Universitat de Lleida

TRABAJO FINAL DE MÁSTER

Estudiante: **TAN KIN TAT**

Titulación: Master en Ingeniería Informática

Titulo de Trabajo Final de Máster: **Arquitectura Big Data para la ejecución de Reglas de Negocio en Tiempo Real**

Director/a: Esteban Chiner Sanz y Francesc Guitart Bravo

Presentación

Mes: Julio

Any: 2019

TABLA DE CONTENIDO

1 INTRODUCCION 2
1.1 Reglas de Negocio en Entornos bancarios 9
1.2 Procesado de datos y gestión de eventos en entornos Big Data 11
1.3 Diferencia entre Batch Processing y Stream Processing 14
1.4 Justificación e interés del tema 15
1.5 Vinculación del tema elegido con las competencias del Máster 19
1.6 Estructura 20
2 OBJETIVOS DEL TRABAJO FINAL DEL MASTER 21
2.1 Objetivos Generales 21
2.2 Objetivos Específicos 21
3 DESARROLLO 22
3.1 Estado del Arte 22
3.2 Herramientas de desarrollo 23
3.2.1 Drools 23
3.2.1.1 Drools API (KIESession) 24
3.2.2 Apache Flink 24
3.2.3 Apache Kafka 25
3.2.4 Apache Kafka Streaming 27
3.2.5 Comparación entre Kafka Streaming y Flink 27
3.2.6 Ejemplos de Proyectos 30
3.2.6.1 Análisis de una integración un motor de reglas, KafkaStream y IoT 30
3.2.6.2 Análisis de una arquitectura aplicando inferencias paralelas con Kafka 32
3.3 Prueba de ejecución con Kafka Streaming 33
3.3.1 Contador de palabras con Kafka Stream 35
3.3.2 Prueba de ejecución con Flink 38
3.4 Propuesta Técnica 41
3.4.1 Datos de Entrada 42
3.4.2 Definición de las reglas de negocio 44
3.4.3 Drools reglas de negocio (.drl) 46
3.4.4 Aplicación de reglas por tabla de decisión (.xls) 48
3.4.5 Resultados 48
3.4.6 Procesamiento 55
4 CONCLUSIONES 67
5 TRABAJO FUTURO 68
6 REFERENCIAS BIBLIOGRÁFICAS 70
7 ANEXO 74

RESUMEN

Actualmente con el aumento de los datos en general, la capacidad de producir información ha avanzado en los últimos años. De acuerdo con SINTEF[1] (Stiftelsen for Industriell og Teknisk Forskning) todos los días creamos 2,5 trillones de bytes en datos por lo cual el 90% de los datos del mundo se han creado solo en los últimos dos años. Según IBM[2], el 90% de todos los datos que generan en los dispositivos conectados como *smartphones*, tabletas, vehículos y electrodomésticos nunca se analiza y el 60% de estos datos empiezan a perder valor en cuestión de milisegundos[3]. El procesamiento en tiempo real es de total importancia en las empresas para identificar a través de análisis de datos, un seguimiento de valores para cualquier toma de decisiones. Con la evolución de las herramientas de comunicación e información todo está conectado entre diferentes tipos y fuentes de información (dispositivos móviles, sistemas inteligentes y Internet de las Cosas [*Internet of Things o IoT*]). Muchas empresas han visto la necesidad de conectar estas fuentes de datos en tiempo real para poder adaptar su operativa de negocio y de este modo procesar y dar respuesta de una manera instantánea. Principalmente, el sector financiero tiene la necesidad de crear productos y servicios para ofrecer para los clientes estrategias innovadoras con una atención personalizada, para atraer satisfacción y fidelización. Un ejemplo de éxito fue planteado por BBVA, que puso en marcha en 2014 la BBVA Data & Analytics, una división de 50 personas, desarrollaron una aplicación llamada *Commerce360*, una herramienta web de inteligencia de negocio que era capaz de estudiar formas de navegación de los usuarios para mejorar y simplificar los procesos[4]. Otro factor de preocupación en las empresas es reducir el tiempo de mantenimiento e inconsistencias en sus proyectos, debido los esfuerzos y costos que son necesarios para realizar el desarrollo. Actualmente muchas empresas se apoyan en sistema de gestión de las reglas de negocio debidos los grandes esfuerzos en el mantenimiento y desarrollo. Los cambios constantes en las reglas de negocios variables, afecta al departamento de

[1] https://www.sintef.no/en/
[2] https://www.ibm.com/
[3]https://www.fundacionbankinter.org/documents/20183/42758/Publicaci%C3%B3n+Big+data/cc4bd4e9-8c9b-4052-8814-ccbd48324147
[4] http://www.expansion.com/economia-digital/companias/2017/04/16/58ecfc32468aeb1c7d8b45d6.html

TI que la gran mayoría las veces requiere un esfuerzo para investigar y mantener la compatibilidad del sistema.

Este trabajo tiene como foco en definir una arquitectura de procesamiento de datos en tiempo real, mejorando la satisfacción de los clientes, identificando sus necesidades en tiempo real, aplicando reglas de negocio a fin de reducir la dependencia de los departamentos de información y negocios.

ABSTRACT

Currently with the increase in data in general, the capacity to produce information has advanced in recent years. According to SINTEF (Stiftelsen for Industriell og Teknisk Forskning) every day we create 2.5 trillion bytes of data, so 90% of the world's data has been created only in the last two years. According to IBM, 90% of all data generated in connected devices such as smartphones, tablets, vehicles and appliances is never analyzed and 60% of this data begins to lose value in a matter of milliseconds. Real-time processing is of utmost importance in companies to identify, through data analysis, a tracking of values for any decision making. With the evolution of communication and information tools everything is connected between different types and sources of information (mobile devices, intelligent systems and Internet of Things [Internet of Things or IoT]). Many companies have seen the need to connect these data sources in real time to adapt their business operations and thus process and respond in an instant. Mainly, the financial sector has the need to create products and services to offer clients innovative strategies with personalized attention, to attract satisfaction and loyalty. An example of success was raised by BBVA, which launched in 2014 BBVA Data & Analytics, a division of 50 people, developed an application called Commerce360, a web tool for business intelligence that was able to study ways of navigating the users to improve and simplify processes. Another factor of concern in the companies is to reduce the maintenance time and inconsistencies in their projects, due to the efforts and costs that are necessary to carry out the development. Currently many companies rely on management system of business rules due to the great efforts in the maintenance and development. The constant changes in the variable business

rules affect the IT department, which most times requires an effort to investigate and maintain the compatibility of the system.

This work is focused on defining an architecture of data processing in real time, improving customer satisfaction, identifying their needs in real time, applying business rules in order to reduce dependence on information and business departments.

PALABRAS CLAVES

Internet of Things, IoT, Business Rules Management System, Stream Processing, Big Data

KEY WORDS

Internet of Things, IoT, Business Rules Management System, Stream Processing, Big Data

INDICES DE ILLUSTRACIONES

Ilustración 1 - Iniciativas en el ámbito del análisis y conocimiento del cliente. 4
Ilustración 2 – Ecosistemas de colaboración y a empresas externas para extraer valor de la información de los datos del cliente. 5
Ilustración 3 - Áreas de riesgo que las organizaciones enfrentarán en los próximos tres años. 7
Ilustración 4 – Preferencias de las empresas en utilización de herramientas de gestión de riesgos. 8
Ilustración 5 – Total de pérdidas por fraudes financiero divididos por tipo en 2016. 9
Ilustración 6 - Representación de los 5 v's del Big Data. 12
Ilustración 7 – Representación de la diferencia entre Proceso Batch y Proceso tiempo real. 14
Ilustración 8 - Informe número de usuários de internet, telefonía móvil y redes sociales. 22
Ilustración 9 – Kafka Arquitectura, Tópicos, Productores y Consumidores. 26
Ilustración 10 - Representación arquitectura por Dumoulin, Mathie. 31
Ilustración 11 - Representación de la arquitectura de aplicación de motor de reglas en Kafka. 33
Ilustración 12 - Interfaz web de las ejecuciones del Flink. 39
Ilustración 13 - Arquitectura integración Kafka Stream aplicación Drools con regla de negocio. 42
Ilustración 14 - Arquitectura definiendo las reglas de negocio de la aplicación 45
Ilustración 15 - Datos de clientes seleccionados como entrada del tópico cliente. 50
Ilustración 16 - Datos de entrada para Client Contacto. 52
Ilustración 17 - Datos de entrada para Cliente Económico. 54
Ilustración 18 - Representación del envío de mensajes a través de los resultados del procesado. 65
Ilustración 19 - Representación del coreo enviado a los resultados del procesado. 65

INDICES DE TABLAS

Tabla 1 - Transformación de los datos de entrada. 44

INDICES DE CODIGOS

Código 1 - Comando para arrancar del servidor Zookeeper pruebas con el Kafka Streaming. 34
Código 2 - Resultado del arranque del servidor Zookeeper pruebas con el Kafka Streaming. 34
Código 3 - Comando para arrancar del servidor Kafka pruebas con el Kafka Streaming. 34
Código 4 - Resultado del arranque del servidor Kafka pruebas con el Kafka Streaming. 35
Código 5 - Creación de un tópico llamado streams-plaintext-input. 35
Código 6 - Ejecución del ejemplo de WordCount propio del Kafka. 36
Código 7 - Ejecución del producer enviando datos de entrada. 37
Código 8 - Mostrando el contador de palabras de salida. 37
Código 9 - Directorio no cual se encuentra el Flink instalado por el Homebrew. 38
Código 10 - Inicia el clúster del Apache Flink. 38
Código 11 - Iniciar un servidor conectando en la puerta 9000 para envío los datos de entrada. 39
Código 12 - Ejecución del contador de palabras consumiendo los datos de entrada de la puerta 9000. 40
Código 13 - Resultado del contador de palabras por Apache Flink. 40
Código 14 - Definición de las reglas de negocio (perfiles) 47
Código 15 - Definición de la regla para contacto del cliente. 48
Código 16 - Comando de verificación del tópico datos clientes. 51
Código 17 - Resultado de envío de los registros datos clientes. 52
Código 18 - Comando de verificación del tópico datos clientes contacto. 53
Código 19 - Resultado de envío de los registros datos clientes contacto. 54
Código 20 - Comando de verificación del tópico datos clientes económico. 55
Código 21 - Resultado de envío de los registros datos clientes económicos. 55
Código 22 - Comando visualización del filtro estudiantil. 57

Código 23 - Resultado filtro estudiantil. 57
Código 24 - Comando visualización del filtro Jubilados. 58
Código 25 - Resultado filtro jubilados. 58
Código 26 - Comando visualización del filtro adultos. 58
Código 27 - Resultado filtro adultos 59
Código 28 - Comando visualización del contador de trabajos. 60
Código 29 - Resultado visualización del contador de trabajos. 61
Código 30 - Comando visualización de los contactos para volver a contactar. 61
Código 31 - Resultado de los contactos para volver a contactar. 62
Código 32 - Comando visualización del contador de duración. 62
Código 33 - Resultado contador de duración. 63
Código 34 - Comando visualización del contador variación en el saldo del cliente. 63
Código 35 - Resultado contador variación en el saldo del cliente. 65

LISTA DE ACRÓNIMOS

Acrónimo	Definición
API	Application Programming Interface
BRMS	Business Rule Management System
CEP	Complex Event Processing
EIU	Economist Intelligence Unit
IoT	Internet of Things
ML	Machine Learning
SINTEF	Stiftelsen for industriell og teknisk forskning
TI	Tecnologías de la información

1 INTRODUCCION

En los últimos años el sector bancario ha tenido una transformación con la digitalización de los datos;

Según Teresa Andrés[5], con la incorporación de las nuevas funcionalidades asociados a los clientes, se han convertido los canales más rápidos para realizar trámites bancarios, permitiendo la gestión bancarias sin necesidad de desplazamientos ni agendamientos de horarios. El 85% de las transferencias bancarias son digitales (36,3 son vía móvil), implica ahorrar exhaustivas colas para ser atendido en los bancos. Los 48,6% de las peticiones por certificados, 47% son tramitados digitalmente. El motivo del incremento de la digitalización es la practicidad en efectuar una transferencia o un pago bancario en pocos minutos, sin necesidad de desplazar a un cajero automático, o en un banco.

Debido a este proceso de digitalización, muchas entidades han aplicado una enorme inversión en Big Data debido a la cantidad de información que se obtiene a través de los mercados financieros, bolsa de valores, redes sociales y los más importante, los clientes. Las tecnologías Big Data permiten entender, actuar y distinguir a los clientes estratégicamente para optimizar sus necesidades. Existen tres funciones principales del Big Data en el sector bancario (ARCE MARTÍNEZ, DANIEL, 2016):

- **Marketing:** La necesidad de conocer a los clientes utilizando su inmensa cantidad de información relacionados de cada respectivo cliente. Resulta necesario utilizar herramientas cada vez más sofisticadas para encontrar maneras de recomendar y personalizar ofertas comerciales, de acuerdo sus necesidades, capacidades, optimizando la venda de sus productos financieros de forma individual.

  El marketing es siempre un aspecto clave en cualquier compañía que sea de una entidad bancaria o no. Imaginamos como un paraguas bajo el que se encuentran las diferentes acciones llevadas a cabo por las empresas a la hora de captar nuevos clientes, como fidelización y retención a los ya existentes. Es de gran importancia conocer el cliente a la hora de definir y

[5] https://www.bbva.com/es/operaciones-bancarias-digitales-crecen-bbva/

desarrollar cualquier aspecto relacionado con la entidad. El Big Data ayuda a analizar los datos y permitir optimizar la segmentación de los clientes, mejorando la gestión de las campañas de marketing y monitorizar la efectividad de las acciones que la entidad bancaria realiza de cara a sus clientes. (MARTIN ORTIZ, NATALIA, 2016)

La segmentación es la subdivisión de un mercado en grupos o segmentos de clientes homogéneos, cuyas respuestas a los esfuerzos del marketing sean iguales o similares durante un determinado periodo de tiempo.

El sector bancario posee un número elevado de clientes, y por ello es uno de los mercados más amplios que existen, lo que da lugar a que la segmentación se considera una herramienta importante. Su objetivo fundamental es analizar la pluralidad de necesidades que surgen dentro del mercado, y su razón es que los consumidores de los productos ofrecidos tiendan a diferenciar los mismos en base a sus expectativas y preferencias. Se centra en las necesidades insatisfechas y en la búsqueda de las posibles oportunidades del mercado, y no en el producto o servicio. Las ventajas es que determina los subgrupos que componen el mercado bancario, desarrolla estrategias más concretas y mejor dirigidas al foco esencial, desarrollar estrategias más apropiadas, mejora control de las tareas relacionadas con el presupuesto de los esfuerzos de marketing. (RIVERA CAMINO, JAIME; DE GARCILLÁN LÓPEZ-RUA, MENCÍA, 2014).

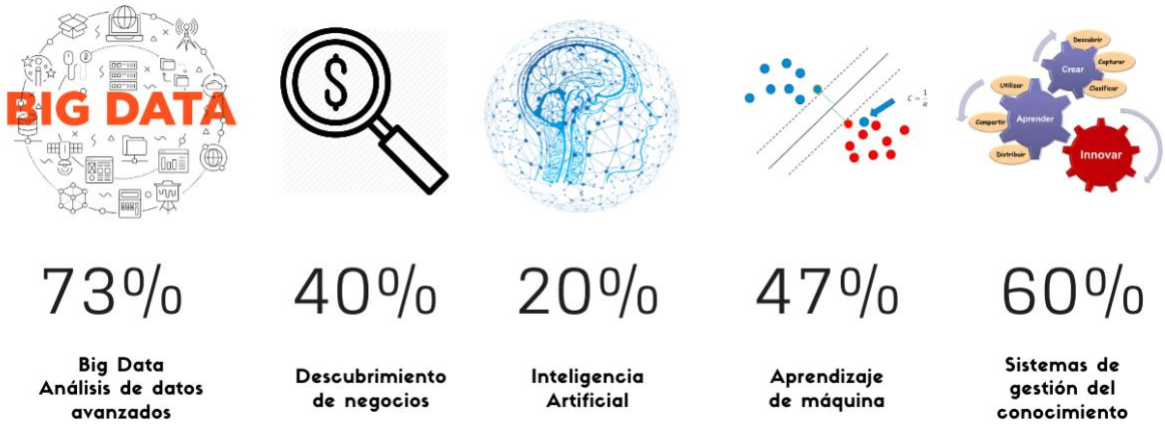
El objetivo de Big Data es utilizar la información para la toma de decisiones, incluso en tiempo real (*streaming*). Las empresas suelen utilizar Big Data para entender el perfil de sus clientes con el objetivo mejorar el proceso de ventas de sus productos y/o servicios vendidos. Big Data permite hacer una segmentación más fina, en principio pudiendo llegar a la individualización, con componentes dinámicos y una segmentación en tiempo real. (GARCÍA MONTALVO, JOSÉ, 2014).

Existe el marketing directo que es la comunicación directa de forma individual con el cliente a través de medios de comunicación masivas, medios tecnológicos, que van dirigidos a una gran cantidad de público. El objetivo

específico es influenciar el cliente a realizar alguna operación para algún interés benéfico con productos específicos. (GONZÁLEZ, TERESA, 2019).

Se debe conocer en profundidad los clientes con objetivo de desarrollar iniciativas en el ámbito del análisis y conocimiento del cliente.

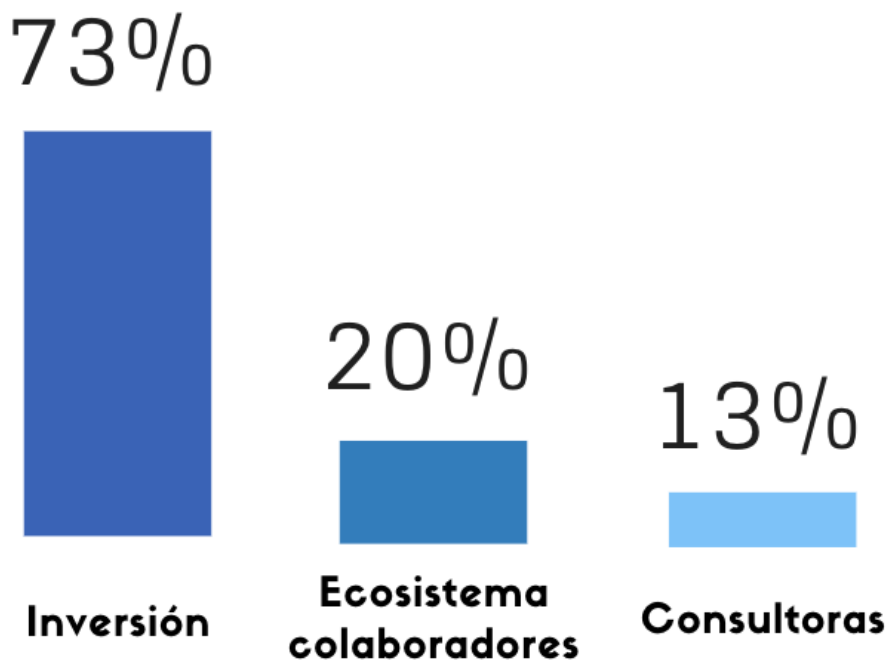
Ilustración 1 - Iniciativas en el ámbito del análisis y conocimiento del cliente.



Fuente: Elaboración propia adaptado de https://www.funcas.es/publicaciones/docs/informe01.pdf.

La Ilustración 1 nos presenta una encuesta realizadas con entidades bancarias. Resulta que el 73% de las entidades actualmente llevan a cabo proyectos de Big Data Análisis de datos avanzados, el 47% aplicando Aprendizaje Automático y el 60% cuenta con iniciativas de Sistemas de gestión del conocimiento. El motivo por el cual las empresas empiezan a aplicar y desarrollar en nuevas tecnologías, es debido la competitividad en fornecer un producto que satisface y atienda mejor el cliente, hay muchas empresas que actúan en el mismo sector por lo tanto es necesario buscar maneras de entender el cliente.

Ilustración 2 – Ecosistemas de colaboración y a empresas externas para extraer valor de la información de los datos del cliente.



Fuente: Elaboración propia adaptado de https://www.funcas.es/publicaciones/docs/informe01.pdf.

La Ilustración 2 representa porcentualmente la realización de las iniciativas seleccionadas anteriormente. Las empresas apuestan en nuevas tecnologías para mejorar el servicio al cliente, como ventas, reducción de costes y etc. Los 73% por ciento de las empresas, consideran que invertir nuevas tecnologías es la salida para encontrar soluciones tecnológicas y desarrollar nuevos productos para entender el perfil de los clientes. Sin embargo, hay colaboradores y consultoras que todavía resisten la introducción tecnológica. (OCAÑA, CARLOS; URÍA, FRANCISCO, 2017)

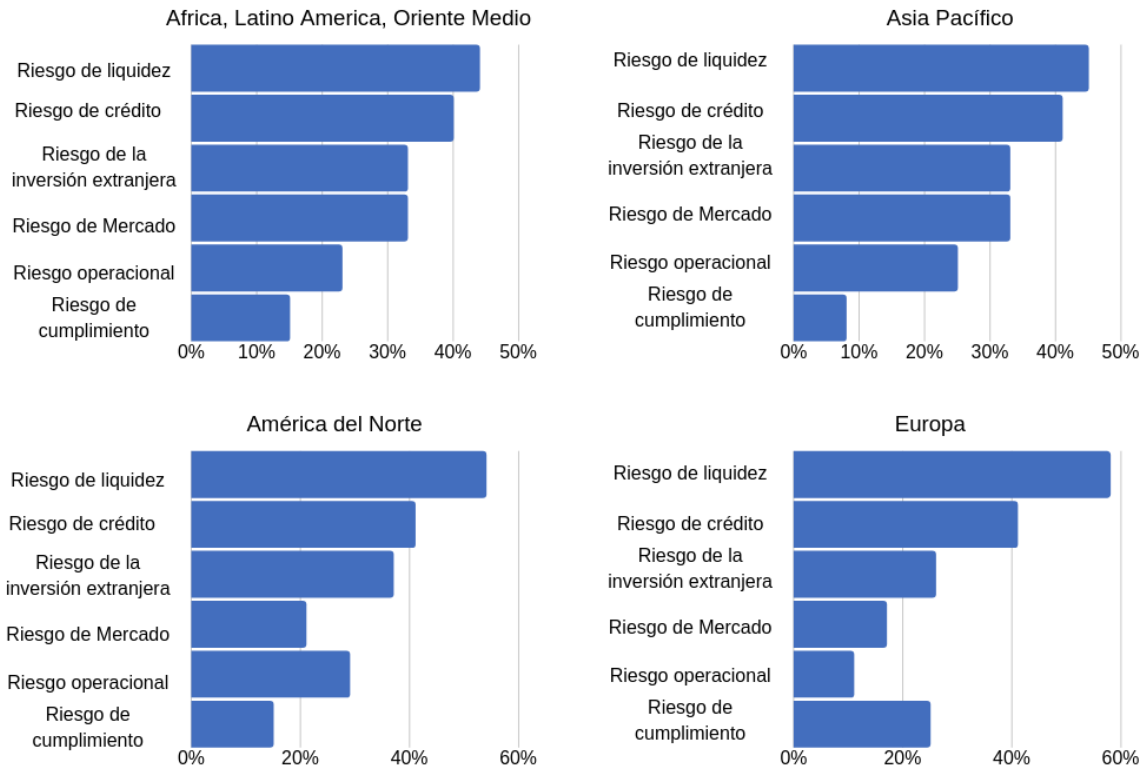
- **Gestión de riesgos:** Se refiere a todos los distintos tipos de riesgos que los bancos enfrentan cuando llevan a cabo sus actividades. Son las medidas que los bancos aplican a las amenazas sobre el sector financiero, adoptando nuevos modelos para mantenerse un paso adelante evitando los errores.

La gestión de riesgos si divide en:

- **Riesgo de crédito** - riesgo que asume cuando concede un préstamo, aval o tarjeta de crédito a sus clientes;
- **Riesgo de mercado** - riesgo que surge ante la eventualidad de incurrir en pérdidas por posibles variaciones adversas en los mercados;
- **Riesgo de liquidez** - Eventual incapacidad de atender los compromisos de pago por parte de las entidades de créditos;
- **Riesgo exterior** - la dificultad de los clientes de determinados países extranjeros de atender sus obligaciones de pago de deudas;
- **Riesgo operacional** - posibilidad de sufrir pérdidas por fallos en procesos o sistemas internos;
- **Riesgo de reputación** - pérdida de imagen y credibilidad de las entidades financieras por parte de los clientes, accionistas y empleados afectando en mantener las relaciones comerciales;

El EIU (*Economist Intelligence Unit*) realizó una encuesta en cuatro regiones, los encuestados provenían de tres tipos de instituciones: 43% bancos comerciales y los 57% divididos entre bancos minoristas y los de inversión. Los tres están preocupados por la liquidez y el riesgo de crédito que otros tipos de riesgo Ilustración 3.

Ilustración 3 - Áreas de riesgo que las organizaciones enfrentarán en los próximos tres años.



Fuente: Elaboración propia adaptado de
https://eiuperspectives.economist.com/sites/default/files/RetailBanksandBigData.pdf.

Observamos en la Ilustración 3 que en todos los continentes el riesgo que las empresas enfrentarán son la liquidez y el crédito. El motivo del riesgo de liquidez es el principal sector es debido la preocupación de que la institución no tenga capacidad de fondear incrementos en sus activos o cumplir con sus obligaciones, sin incurrir en costos financieros fuera del mercado. El riesgo de crédito es debido cuando tras la venta a plazo de un producto, el cliente termina dejando de pagar, es necesario tener soluciones en marcha en estas situaciones. Con el Big Data es posible prever pérdida o incremento de valor de productos para ser aplicado en el mercado, o identificación de los clientes de acuerdo con su perfil de acuerdo con la inmensa cantidad de dato.

Ilustración 4 – Preferencias de las empresas en utilización de herramientas de gestión de riesgos.

Fuente: Elaboración propia adaptado de https://eiuperspectives.economist.com/sites/default/files/RetailBanksandBigData.pdf.

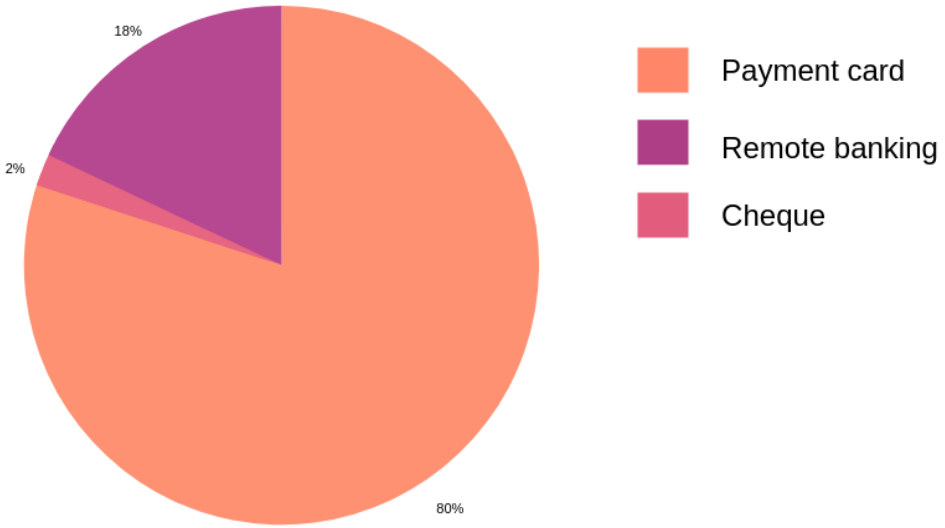
La Ilustración 4 demuestra que las herramientas y técnicas de gestión de riesgo son fundamentales en las organizaciones. La gran mayoría (81%) están utilizando en sus negocios, ven la necesidad de gestionar problemas que consecuentemente pueden acontecer.

- **Gestión de fraudes:** la gestión de fraudes a través de Big Data permite implementar medidas para la detección de actividades fraudulentas, lo que les permitirá reducir las pérdidas asociadas a fraudes, medios de pagos y prevención de blanqueo de capitales o financiación del terrorismo.

  Según el informe *Financial Fraud UK*[6], en 2016, las pérdidas por fraude financiero en las tarjetas de pago, la banca remota y los cheques dieron como resultado 768,8 millones de libras, un aumento del 2% en comparación con 2015. El fraude evitado ascendió a 1.380 millones de libras en 2016 (Ilustración 5). (KOTESHOV, DMITRI, 2017).

[6] https://www.financialfraudaction.org.uk/fraudfacts17/assets/fraud_the_facts.pdf

Ilustración 5 – Total de pérdidas por fraudes financiero divididos por tipo en 2016.



Fuente: Elaboración propia adaptado de https://www.elinext.com/blog/fraud-management-detection-and-prevention-in-banking-industry/.

Es necesario crear opciones de uso de software de analice de datos para garantizar detección de fraude para evitar futuros problemas a los clientes. Las nuevas herramientas de procesamiento nos posibilitan por ejemplo identificar algún fraude de acuerdo con sus movimientos de manera individual y tiempo real.

1.1 Reglas de Negocio en Entornos bancarios

Los mercados financieros sufren con la presión normativa y la fuerte competencia generada por la globalización de los mercados financieros. Existen muchas medidas legislativas puestas en marcha por la Unión Europea que afecta en aspectos la actividad financiera, desde la gestión de activos de crédito hasta los mercados de instrumentos financieros. Las empresas necesitan estar siempre en la parte más alta, competiendo para proporcionar mejores productos, para eso es necesario adaptar de manera rápida para renegociar los contratos con sus clientes por motivos de fidelización. Deben ser sustituidos los sistemas rígidos de pagos para sistemas más flexibles que permita adaptar las políticas de precios ofreciendo una personalización para cada cliente. Implica en utilizar

conjuntos de funciones definidas y reutilizables, como de validación, cálculos en general. La utilización de una arquitectura orientada a servicios es el objetivo de domar la complejidad, porque permite construir servicios de decisión basados en reglas de negocio. Según cambien las circunstancias, el banco podrá revisar y actualizar fácilmente sus políticas de riesgo aplicadas en el proceso y volver a procesarlo de manera automática[7].

Para Ross las reglas de negocio definen como la política que rige el comportamiento de la empresa y la distingue de las demás, también como una regla como una 'política o práctica comercial operativa discreta' una declaración declarativa expresada en términos 'no técnicos' (ROSS, RONALD; 1987). Halle ve las reglas como condiciones que 'gobiernan un evento de negocios para que ocurra de una manera que sea aceptable para el negocio, los eventos deben verse como eventos que resultan en una actualización de una base de datos. (HALLE, VON; 2002). Para Morgan define una regla de negocio como una declaración compacta sobre un aspecto de un negocio que puede expresarse en términos que pueden ser directamente relacionado con el negocio, utilizando un lenguaje simple e inequívoco que sea accesible para todas las partes interesadas: propietario de negocios, analista de negocios, arquitecto técnico, etc. (GRAHAM, IAN; 2006).

Los sistemas de BRMS (Business Rule Management System) viene creciendo a medida que el negocio requiere más flexibilidad y velocidad a la hora de un cambio de reglas de negocio; permite la administración y ejecución de reglas de negocio sean escritas por los usuarios de negocio sin la participación del equipo técnico. (MARTIN IBARRA, GONZALO; BAZÁN, PATRICIA; 2013).

Es diferente el desarrollo de los proyectos tradicionales, la implementación de un sistema BRMS es necesario un fuerte compromiso desde el área técnica y de negocios. El área técnica implica la diferencia de ciclo de vida y metodologías de desarrollo de un proyecto tradicional, porque se amplía más roles y actores en su ejecución. El área de negocios cumple un papel fundamental para el éxito del proyecto.

[7] http://gestionpyme.com/las-reglas-de-negocio-en-el-nuevo-entorno-financiero/

1.2 Procesado de datos y gestión de eventos en entornos Big Data

Big Data se define como la gestión y el análisis de enormes volúmenes de datos que no pueden ser tratados de manera convencional porque superan los límites y capacidades de las herramientas habituales para la captura y procesamiento de datos. Son activos de información[8], caracterizado por su gran volumen, velocidad y variedad, y esto hace que demanden soluciones innovadoras y eficientes de procesado para la mejora del conocimiento y toma de decisiones en las organizaciones. (LÓPEZ GARCÍA, DAVID, 2013).

Los requisitos entre trabajar con un conjunto de datos grandes o pequeños son los mismo. La diferencia es la escala masiva, velocidad de ingestión y procesamiento, y las características que deben ser tratados en cada etapa del proceso es el grande desafío para diseñar nuevas soluciones. Las características que hacen Big Data ser diferentes de otros flujos de procesado de datos son:

- Volumen: La gran escala de la información que son procesada se define los sistemas de Big Data. Conjuntos de datos superiores que los tradicionales que exige una mayor reflexión en cada etapa del ciclo de vida del procesamiento y almacenamiento. Con el problema de las capacidades de procesamiento distribuidos (clústeres), se convierte en un desafío de agrupar, asignar y coordinar recursos de grupos de computadoras para solucionar los problemas.

- Velocidad: La frecuencia los datos fluyen al sistema desde múltiples fuentes, y es necesario procesar en tiempo real para obtener información a la toma de decisiones. Los datos se agregan, enviadas, se procesan y se analizan constantemente para mantenerse al día con la afluencia de información mostrando información valiosa cuando es más relevante.

- Variedad: Hay una gran variedad de fuentes de información que pueden ser útiles en el procesamiento. Los registros de las aplicaciones, servicios de

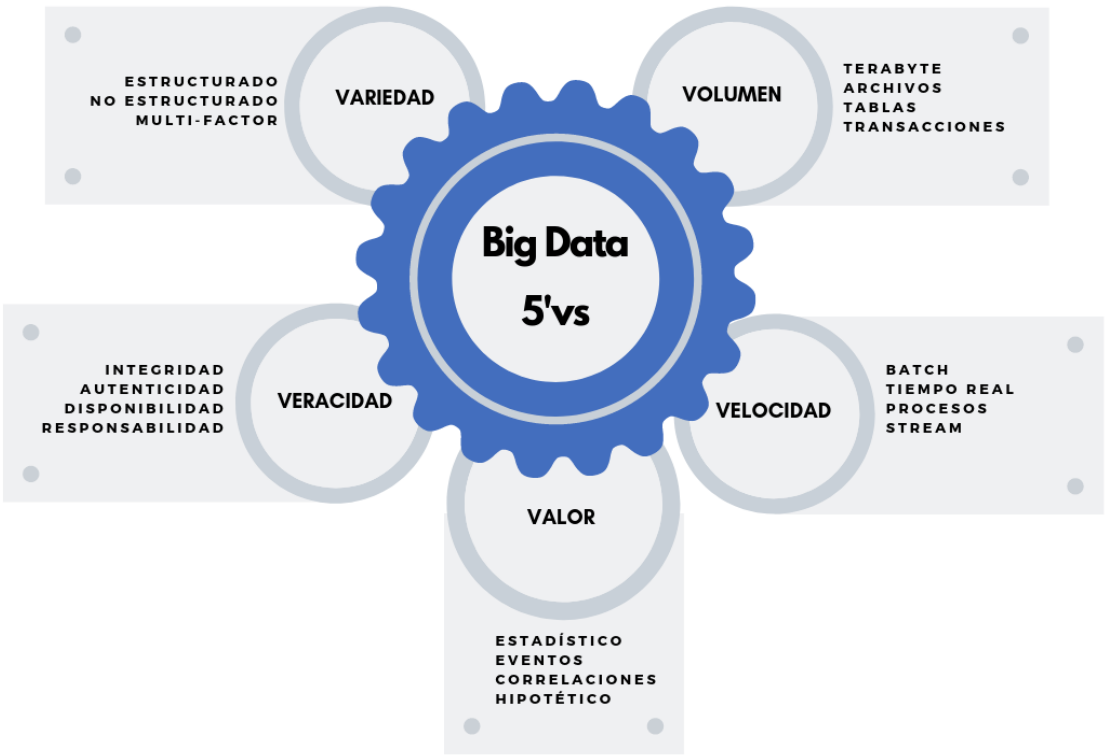
[8] Es algo que una organización valor y por lo tanto debe proteger. Son ficheros, bases de datos, contratos, documentaciones, manueles, etc.

feeds de las redes sociales, APIS (Application Programming Interface) externas, sensores de dispositivos físicos, etc.

- Veracidad: La variedad de fuentes y la complejidad del procesamiento nos llevan a desafíos en la evaluación de la calidad de los datos. El grado de confianza que se establece sobre los datos.

- Valor: El mayor de los desafíos es entregar valor, los sistemas y procesos implementados son lo suficiente complejos como para que el uso de los datos y la extracción de valores reales pueda extraer conocimiento e información útil.

Ilustración 6 - Representación de los 5 v's del Big Data.



Fuente: Elaboración propia adaptado de https://static1.squarespace.com/static/581fbf655016e1dae7ae03a2/t/58beff5e44024341e963ce8b/1488912240459/The+5+Vs+of+Big+Data?format=750w.

La diferencia entre Data Streaming y Stream Processing es que Data Streaming es apenas el concepto de miles de fuentes de datos (aplicaciones móviles o web, compras electrónicas, redes sociales, actividades de jugadores online, etc), que se envían simultáneamente información a punto a ser procesados, en cambio Stream Processing es la tecnología capaz de procesar los datos en tiempo real.

Antedicho, el *Data Streaming* es un método de manejo de datos en el que la información se analiza y organiza a medida que son producidos, una vez que los datos son generados a partir de un evento, se procesan de manera secuencial o en ventanas de tiempo utilizados para ampliar la variedad de tipos de análisis, correlaciones, agregaciones, filtrado, muestreo y etc.

El Stream Processing nos permite procesar los datos en tiempo real a medida que llegan las informaciones detectando rápidamente las condiciones dentro de un período de tiempo pequeño desde el momento que recibimos los datos, nos permite enviar datos a las herramientas de análisis tan pronto como se generan y obtienen resultados de análisis instantáneos. Es útil por ejemplo para tareas como la detección de fraude, si procesa los datos de las transacciones en forma continua, puede detectar anomalías que señalan un fraude en tiempo real y luego detener las transacciones fraudulentas antes de que se completen.

En contrapartida el Batch Processing (procesamiento por lotes) es el procesamiento de bloques de datos históricos, que se han almacenado durante un período de tiempo, por ejemplo, el procesamiento de todas las transacciones realizadas semanalmente o diariamente por una importante empresa financiera para realizar análisis. (VASEEJARAN, GOWTHAMY, 2017).

El Complex Event Processing[9] es un procesamiento de eventos que combina datos de múltiples fuentes para inferir eventos y patrones que sugieren circunstancias más complicadas. La característica principal es que pueden identificar eventos e informar en tiempo real.

[9] https://en.wikipedia.org/wiki/Complex_event_processing

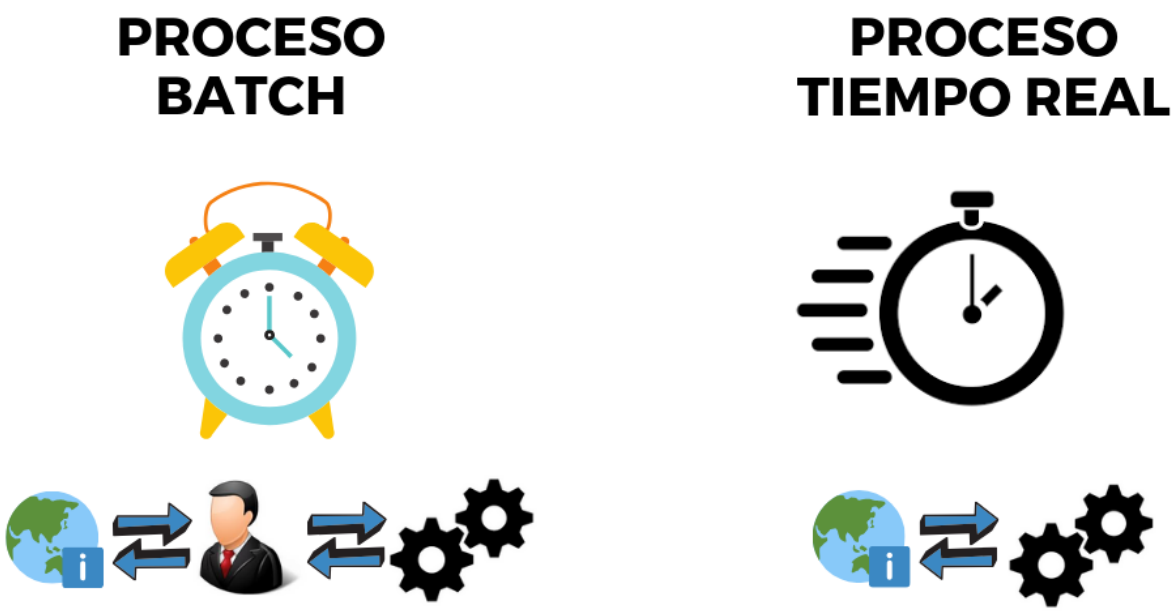
1.3 Diferencia entre Batch Processing y Stream Processing

La diferencia entre Batch Processing y Stream Processing, es que el Batch Processing procesa sobre a la mayoría de los datos, y el stream processing procesa los datos en una ventana móvil o en el registro más reciente.

Ejemplos de Batch Processing, compensación de cheques, transacciones de tarjetas de crédito, generación de cuentas, generación de informes financieros, procesamiento de imágenes, procesamiento de entrada y salida en el sistema operacional.

Ejemplos de procesamiento en tiempo real: Reserva de boletos en línea, Procesamiento de video en vivo, Procesamiento en evento en vivo, Terminal de punto de venta, Transacciones de dinero en línea, Procesamiento en el Sistema Operativo, Transacciones en cajeros automáticos del banco, Servicio al cliente, Sistema de radar, Previsión meteorológica, Medición de temperatura. (Ilustración 7)

Ilustración 7 – Representación de la diferencia entre Proceso Batch y Proceso tiempo real.



*Fuente: Elaboración propia adaptado de https://cdn-images-1.medium.com/max/1600/1*32pf9w6Sr2Q-79iiWlf4-A.jpeg.*

La diferencia entre Event Streaming Processing con el Complex Event Processing es que el Event Streaming Processing es la captura y el procesamiento de una gran cantidad de eventos en una ventana temporal concreta. Podemos definir como una secuencia de eventos, del mismo tipo, ordenados en el tiempo. Por su parte, el Complex Event Processing tiene por meta la captura y procesamiento de eventos de diferente tipo de una forma desordenada en la llamada 'nube' de eventos. Es decir, que contiene muchas corrientes de eventos. Por ejemplo, un sistema financiero que tiene por objetivo el procesamiento en tiempo real de las acciones y de las noticias asociadas a las compañías que cotizan en las bolsas de Madrid. Para Event Streaming Processing podríamos obtener la cotización media que ha tenido una acción en una corriente de eventos concreta, para Complex Event Processing, podríamos definir patrones complejos que correlacionen la publicación de una noticia con la cotización de una compañía para tomar decisiones al respecto, una regla podría ser si aparece una noticia de una compañía y en un intervalo temporal determinado la cotización de la acción sufre una brusca bajada entonces vender automáticamente las acciones de dicha empresa. (AYLLÓN, VICTOR; M. REINA, JUAN, 2008).

1.4 Justificación e interés del tema

Actualmente las empresas utilizan tablas relacionales para el almacenamiento de los datos de los clientes. En la actualización del saldo del cliente, por ejemplo, es necesario una constante actualización en la tabla correspondiente con el registro del cliente. Siempre queda la última foto de la información del cliente reflejados en la base de datos. El objetivo de una arquitectura de streaming es utilizar los logs de actualizaciones para encontrar soluciones para los problemas.

Por ejemplo:

Imaginamos que un cliente, se ha efectuado 3 (tres) transacciones en diferentes lugares del mundo, en relación geográfica es imposible que una persona esté en

tantos lugares en un periodo pequeño de tiempo, implica que es una tarjeta clonada. Una arquitectura streaming combinando con un motor de reglas facilita el sector de negocios para ser más flexibles en cambios de reglas de negocios ahorrando el tiempo de desarrollo de la equipe técnica.

Los modelos de negocio son modelos esquemáticos que describen la manera en que las empresas crean y producen valor para sus clientes. El concepto de modelo comprende como el producto, el cliente y el mercado, por lo cual el papel de la empresa dentro de la cadena es el motor económico que le permite alcanzar sus objetivos de rentabilidad y crecimiento. Un modelo de negocio es un plan de acción estructurado que tiene como objetivo aportar orden y disciplina al caótico proceso de creación, expansión y gestión de un negocio. Es el modo que la empresa crea valor y obtiene ingresos y beneficios, lo que se define a través de tres elementos fundamentales: un modelo de creación de valor, un modelo de beneficios y la lógica de los negocios.

La lógica del negocio es la organización de cómo las empresas van a cumplir sus objetivos de beneficios y crecimiento. Existen muchos arquetipos en los modelos de negocio, cada uno con una lógica propia. Para explicar utilizaremos tres de estos arquetipos.

1. El primero (front-end) es basado en la proximidad al cliente en que la empresa encuentra soluciones con el foco al cliente, centradas en el extremo final de la cadena de valor, es la lógica que tiene una conexión directa con el cliente que tiene una relación de aprendizaje continuo con los clientes, que capta información acerca de sus hábitos, preferencias e informaciones para personalizar los productos y servicios.
2. El segundo (back-end) es focaliza en la excelencia de las operaciones, que está en la parte inicial de la cadena, el objetivo es crear procesos para mejorar, agilizar y disponibilidad el servicio para atender mejor el cliente.
3. El tercero es la parte de coordinación de la cadena de valor, coordina los elementos iniciales y finales extremos de la cadena de valor, tiene el objetivo de facilitar transacciones o interacciones entre los usuarios.

En el mercado de empresarial los tipos más comunes de aplicaciones gira en torno de la aplicación de algún tipo de lógica empresarial a los datos generados desde diversos aspectos del negocio. (MENDELSON, HAIM, 2015).

La cantidad de información que se genera por segundo a los tiempos actuales es gigantesca, las empresas están siempre buscando manera de desarrollar aplicaciones empresariales modernas para conectar cada vez más los tipos de fuentes de datos, cualquier tamaño y cantidad de datos, resultando información más confiable. En una aplicación empresarial el ciclo de desarrollo de una aplicación personalizada muchas veces tiene un plazo de un año o más, a medida que cambian las necesidades y las condiciones de negocio se quedan menos atractivas y la aplicación se vuelve obsoleta antes de que se ponga en producción.

Las organizaciones que utilizan un gran volumen de datos necesitan utilizar tecnologías de Big Data para el procesamiento, así que abre una nueva clase de dificultades que han hecho el costo de desarrollar aplicaciones empresariales a escala que sea extremadamente costoso, poniendo siempre barreras en infraestructura de TI con modificaciones y alteraciones, coste de tiempo y financiero que tendrá la empresa.

Por este motivo el área de negocio necesita ejecutar y modificar la lógica empresarial de acuerdo con las necesidades de la empresa sin el demasiado esfuerzo de la infraestructura de TI (tecnología de la información), permitiendo recopilar a través de una variedad de fuentes con escaladas muy grandes en tiempo real, ejemplos como datos bancarios, señalizaciones de tránsitos, dispositivos conectados a IoT (Internet of Things). (DUMOULIN, MATHIEU, 2017)

Actualmente las empresas tienen una acumulación de TI de los cambios solicitados que pueden tardar meses o años en entregarse, un BRMS ayuda el cambio de la lógica empresarial más ágil y más sencilla que el desarrollo tradicional. (MURALI KRISHNA, KUMAR BARUAH, PALLAV, 2018)

Tiene como objetivo ayudar a automatizar las decisiones que rigen en las políticas de una empresa, reduce la necesidad de procesamiento manual.

Cuando las decisiones de negocios toman demasiado tiempo el cliente sufre en su servicio. En el pasado, las compañías de seguros de salud tardaban un promedio de dos semanas en procesar nuevas solicitudes de seguro. En la actualidad las compañías pueden procesar solicitudes en cuestión de días, o incluso horas, mediante el uso de motores de reglas empresariales. El uso de las reglas implica que los clientes tienen interacciones de servicio al cliente sencillas y sin complicaciones. Mejora de las ventas, beneficios adicionales de generar lealtad a la marca desde el principio.

Un ejemplo es de la California Association of Realtors, que utiliza los motores de reglas comerciales para ayudar a sus miembros a determinar que formularios necesitaban para cerrar sus acuerdos de bienes. Antes lo hacían una línea directa con abogados para asesorar a los agentes inmobiliarios sobre los formularios requeridos. Se capturó la guía de los abogados en las reglas comerciales y automatizan con el motor de reglas y ofrecer en un sitio web interactivo, pues la aplicación está de total disposición al cliente para acceder 24 horas del día. (PROGRESS, 2015).

El procesamiento streaming analiza y realiza acciones en datos en tiempo real a través del uso de consultas continuas. Streaming Analytics se conecta a fuentes de datos externas, permite aplicaciones integrar datos en flujo de aplicación, o actualizar una base de datos externa con la información procesada. Streaming Analytics es la capacidad calcular constantemente análisis mientras se mueve dentro del flujo de datos, permite la gestión, la supervisión y el análisis en tiempo real de los datos de transmisión en vivo. Implica en conocer y actuar sobre los eventos que ocurren en su negocio en un momento rápido. Los datos analizados son producidos y las empresas deben actuar sobre los datos rápidamente en una pequeña ventana de oportunidad antes de que los datos pierdan su valor. Los datos que pierden su valor dan como costos adicionales (operativos, administrativos, riesgos comerciales, daños a la reputación, posibles acciones legales, reducción de la productividad, incapacidad para tomar decisiones, reducen la ventaja competitiva de una empresa).

Es importante porque nos permite realizar los riesgos antes de que ocurran, puede ayudar a las compañías a identificar nuevas oportunidades de negocios y flujos de ingresos que resulten en un aumento de las ganancias, nuevos clientes

y un mejor servicio al cliente. "*Because data in a Streaming Analytics environment is processed before it lands in a database, the technology supports much faster decision making than possible with traditional data analytics technologies*" - "Debido a que los datos en un entorno de Streaming Analytics se procesan antes de llegar en una base de datos, la tecnología permite una toma de decisiones mucho más rápida de lo que es posible con las tecnologías tradicionales de análisis de datos" dijo Philip Howard[10]. Streaming Analytics ayuda a proporcionar protección de seguridad porque brinda a las empresas una forma rápida de conectar rápidamente diferentes eventos para detectar patrones de amenazas de seguridad y sus riesgos, y para realizar un monitoreo de seguridad de la red y los activos físicos. (FREEMAN, HARRINE, 2016).

1.5 Vinculación del tema elegido con las competencias del Máster

En las empresas el tema de Big Data, datos en tiempo real y reglas de negocio se aplicando a menudo. El Big Data es un tema que las empresas viene adoptando para mejorar el aspecto de estadísticas y entender los clientes a través de la cantidad masiva de datos; las reglas de negocio en optimizar el flujo de acciones tomadas en los productos, estos temas tienen un fuerte vínculo con las competencias del Máster:

- Desarrollo de Software
- Elaboración, planificación estratégica, dirección, coordinación y gestión técnica y económica de proyectos de la Ingeniería Informática
- Dirección técnica de las actuales herramientas de investigación y innovación.
- Capacidad para la aplicación de los conocimientos adquiridos y la resolución de problemas en entornos nuevos o poco conocidos
- Capacidad para utilizar y desarrollar metodologías, métodos, técnicas, programas de uso específico.

[10] https://www.datamation.com/data-center/streaming-analytics-business-value-from-real-time-data.html

1.6 Estructura

El en **primer apartado** se hace una introducción de los conceptos que serán utilizados para el desarrollo del proyecto, teniendo el foco para entornos bancarios que es el real centro del proyecto, explica las definiciones de las reglas de negocios con su real objetivo en los proyectos debido su facilidad de organizar y agilizar las condiciones de negocio desde el punto de vista del sector de negocio, Big Data que es el trabajo con grandes cantidades de informaciones que desde los últimos tiempos se creció muchísimo por la capacidad de generación de dados en cada instante y batch y stream processing que son conceptos de procesamiento en lotes o por tiempo real, todos estos conceptos son de grande importancia en las empresas bancarias de la actualidad.

El **segundo apartado** hace referencia a los objetivos, tanto generales como específicos, que abarca este documento.

El **tercer apartado** es el desarrollo del proyecto, que se encuentra el de estado de arte que son proyectos ya realizados para que sea posible ver el nivel de comprensión recientes de las herramientas utilizados con objetivo de entender y mejorar algo que ya fue realizado. Explicación de las herramientas que se utiliza para tener un nivel de comprensión del comportamiento y el valor de uno a uno.

Se encuentra la propuesta del proyecto es la definición de la arquitectura y desarrollo de acuerdo con el tema definido, la finalidad es demonstrar paso a paso del trabajo desde la obtención de los datos hasta los resultados del procesamiento.

Los **siguientes apartados** contienen una conclusión de los resultados obtenidos de acuerdo con el trabajo realizado y también definición de trabajo futuro que tiene el objetivo de mejorar el trabajo facilitando más el proceso con visión de negocio.

Para finalizar el **último apartado** está las referencias bibliográficas y los anexos utilizadas al largo de todo el trabajo.

2 OBJETIVOS DEL TRABAJO FINAL DEL MASTER

2.1 Objetivos Generales

- Aplicación de conceptos de Big Data
- Utilizar concepto de reglas de negocio para facilitar el cambio en el proyecto la parte del cliente.
- Diseño de estructura de streaming para reglas de negocio

2.2 Objetivos Específicos

- Investigación de los conceptos de Big Data, Streaming de Datos y reglas de negocio
- Importancia en utilizar los conceptos aplicados en empresas bancarias
- Diferencias entre Kafka y Flink
- Desarrollo capa de los productores
- Desarrollo capa del procesamiento
- Desarrollo capa de los consumidores
- Mejoría aplicando tabla de decisiones para mejorar la gestión de parte de cliente.

3 DESARROLLO

3.1 Estado del Arte

Durante las últimas décadas, hubo un crecimiento exponencial en los medios de información.

Ilustración 8 - Informe número de usuários de internet, telefonía móvil y redes sociales.

Fuente: https://s3.amazonaws.com/cdn.wp.m4ecnet/wp-content/uploads/2018/02/31225729/grafica-1-usuarios-de-internet-en-el-mundo-totales.jpg.

Se define el período de la revolución digital o la era de la información, donde los teléfonos móviles están conectados en la red enviando información a todo momento; es común cada persona tener disponible un ordenador portátil, móvil, pudiendo compartir información en cualquier lugar. (Ilustración 8)

La información es de gran importancia, grandes empresas tecnológicas dedican veinticuatro horas diarias enviando y recibiendo datos a cada segundo, generando una cantidad inmensa de datos día a día, que necesita de mecanismo de procesamiento en tiempo real encontrar soluciones más eficaces de los problemas.

Las empresas han tenido que adaptarse a diferentes desafíos para atender esta demanda de información, los principales retos es posibilitar la manipulación, administración, almacenamiento, búsqueda y analice con grandes volúmenes de datos: Big Data.

3.2 Herramientas de desarrollo

Este aparcado tiene el propósito de explicar con más detalles los importantes programas utilizados para el desarrollo del trabajo, para tenernos una pequeña idea de su creación como también el funcionamiento. La comparación entre las herramientas de streaming, características individuales de cada herramienta y ventajas y desventajas, como también explicar algunos ejemplos de proyectos que aplican estas herramientas para tener una idea en general del potencial para que sea posible mejorar y potencializar este proyecto. Las herramientas son: Drools, Apache Flink, Apache Kafka.

3.2.1 Drools

Drools es un BRMS implementado por Red Hat que provee de herramientas para la definición de las reglas. Se encarga en su proceso de compilación, recreación del árbol de flujos/decisiones (basado en un algoritmo de inferencia Rete[11]). Reta es un algoritmo de reconocimiento de patrones no cual se genera un árbol de decisiones. Drools tiene la capacidad de declarar las reglas de forma sencilla a través de la lenguaje de reglas propia del drools (.drl) o de los ficheros excel (tablas de decisiones).

Los procesos de negocios pueden tener varios caminos de ejecución distintos, en muchos casos quien determina el camino puede estar relacionado por una decisión de negocios, por lo tanto, es necesario que una aplicación sea flexible y no rigurosos a punto de requerir demasiado esfuerzo a cambios. Mayoría de las empresas la lógica de negocio está separados en diferentes sitios, aumentando su complejidad de cambios y gestiones. Encuentra se en los códigos de las aplicaciones, hojas de cálculos y muchas veces en las mentes de los expertos en el proyecto. Drools es un

[11] https://es.wikipedia.org/wiki/Algoritmo_Rete

gestor de reglas BRMS que el propósito es de auxiliar para la gestión de las reglas de negocio, centralización y gestión.

La lógica es definida para ser aplicada a alguna regla dentro de un determinado problema de la empresa, las reglas tienen apenas el propósito para representar la lógica del negocio.

La codificación de la lógica de negocio tiene como objetivo de facilitar la comprensión del proyecto, sobre todo para el personal técnico.

3.2.1.1 Drools API (KIESession)

Es la forma más común de interactuar con el motor de reglas del Drools. Permite que la aplicación establezca una conversación iterativa con el motor, donde el estado de la sesión se mantiene a través de las invocaciones. Por lo tanto, es posible enviar objetos para que el motor ejecute las reglas de negocio definidas.

3.2.2 Apache Flink

Apache Flink comenzó como una colaboración de varias universidades europeas en un proyecto de investigación llamado "*Stratosphere: Information Management on the Cloud*[12]", cuando pasó a formar parte de Apache Incubator[13] (incubadora de proyectos de código abierto destinados a convertirse en proyectos Apache). Flink es una plataforma de código abierto de procesamiento en streaming de datos escalables y procesamiento por lotes. Ha sido diseñado para ejecutar cálculos de estado a través de flujos de datos. Es optimizado para procesar flujos de datos ilimitados, como conjuntos de datos acotados de cualquier tamaño, capaz de escalar los cálculos a miles de núcleos, procesar flujos de datos con un alto rendimiento y con baja latencia, operar en un modo de alta disponibilidad sin punto de falla y recuperar aplicaciones con estado de fallas con garantías de consistencia de estado.

La utilización del Apache Flink es de implementar sistemas distribuidos eficientes que respondan rápidos a preguntas computacionalmente complejas (ML - Machine Learning, estadísticas, cálculos) como procesos de limpieza y pre-filtrado sobre cantidades ingentes de información, detección anomalías, sistemas de tiempo real de monitorización alertas, IoT, etc…

[12] http://stratosphere.eu/project/funding/

[13] https://incubator.apache.org/

Algunas de las importantes empresas internacionales que utilizan Flink: Zalando, una de las compañías e-commerce más grandes de Europa que utiliza la herramienta para procesos de monitorización de compras en tiempo real; Kings (empresa creadora de la saga "Candy Crush") utiliza Flink para el procesamiento de más de 20 billones de eventos diarios; el grupo Alibaba (dedicado al comercio electrónico, fundadores de AliExpress y Tobao), utiliza Flink en un clúster de 1.000 nodos (con más de 5.000 cores) como base de un sistema de ranking de búsquedas en tiempo real; la compañía Boygues (líder del sector ingenieril en Francia, teniendo filiales como Telecom o Dragados) dispone de 30 aplicaciones de Flink en producción, las cuales procesan 10 billones de eventos diarios (computacionalmente se estiman unos 2 terabyes).

3.2.3 Apache Kafka

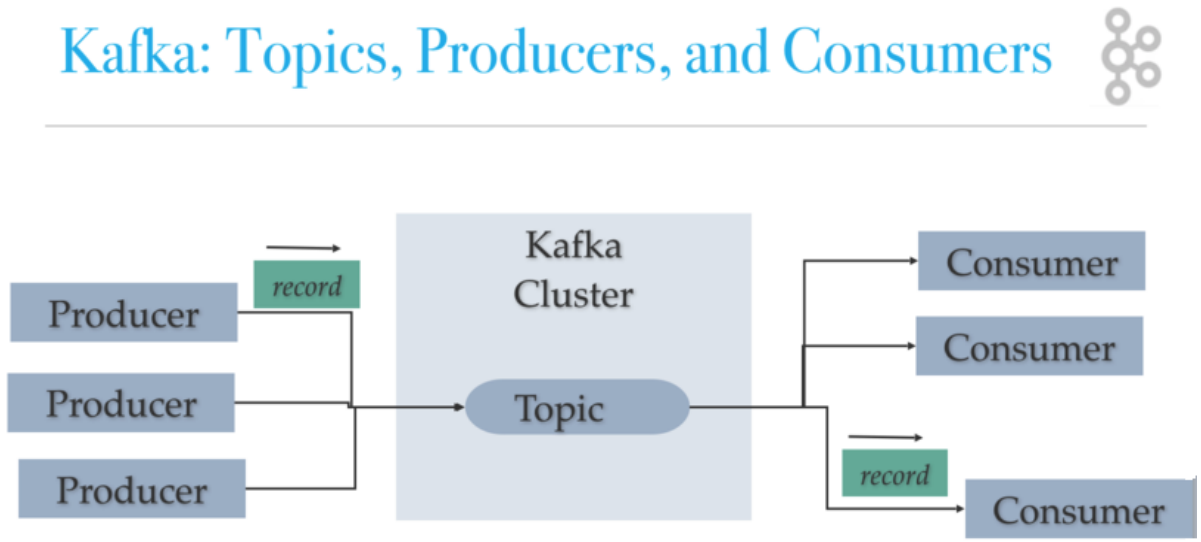
Apache Kafka fue desarrollado alrededor de 2010 en Linkedin por un equipo que incluía Jay Kreps, Jun Rao y Neha Narkhede. Si originó con una propuesta para resolver la ingestión de baja latencia con gran cantidad de datos de eventos de la página web y la infraestructura de LinkedIn que utilizada una arquitectura lambda que aprovechada los sistemas de procesamiento de eventos en tiempo real y Hadoop. Apache Kafka es un sistema que usa las arquitecturas de datos de streaming en tiempo real para proporcionar análisis en tiempo real, un sistema de publish-subscribe que entrega mensajes en orden, persistentes y escalables, los tópicos permiten el consumo masivo paralelo.

Los mensajes escritos en Kafka son persistidos y replicados tolerantes a fallas, los mensajes permanecen alrededor por un período de tiempo configurable. La principal clave de Kafka es el Log, estructura de datos de registros, es una secuencia de inserciones de datos ordenados por el tiempo. Los datos pueden ser cualquier información, tipo o estructura. Simplemente son arreglos de bytes que pueden ser utilizados para almacenar cualquier objeto en cualquier formato, siendo String JSON[14], AVRO[15] etc…

[14] JSON es un formato ligero de intercambio de datos, independiente del lenguaje de programación. La característica envuelve su estructura plana, lectura, escritura y generación simples.

La arquitectura básica del Kafka es organizada por productores, consumidores, tópicos y brokers[16].

Ilustración 9 – Kafka Arquitectura, Tópicos, Productores y Consumidores.



Fuente: http://cloudurable.com/images/kafka-architecture-topics-producers-consumers.png.

- Los productores de mensajes se les llaman publishers y a los consumidores de mensajes se les llama subscribers.
- Los productores son aplicaciones que pueden actuar como una fuente de datos en un cluster en Kafka. Es responsable en publicar mensajes en los tópicos del Kafka.
- Los consumidores actúan en consumir los mensajes en uno o más tópicos para se utilizar en procesamiento o algún tratamiento de datos.
- Los brokers son un sistema simple responsable de mantener los datos siempre publicados. (Ilustración 9)

El clúster Kafka consisten en múltiples brokers para mantener el equilibrio de carga. Los brokers no tienen estado, utilizan ZooKeeper (servicio centralizado para diversas tareas para el mantenimiento de configuración, sincronización distribuida, servicios

[15] AVRO es un sistema de serialización de datos basados en esquemas JSON, permite una representación de estructuras complejas. Los datos son almacenados en formato binario lo que convierte en una herramienta liviana y eficiente.
[16] Una instancia de un servidor Kafka.

de agrupación, etc.) para mantener su estado de agrupación. Una instancia de Kafka broker puede manejar ciento de miles de lecturas y escrituras por segundo pudiendo manejar TB de mensajes sin impacto en el rendimiento.

Kafka tiene una operativa simple, la principal razón por la que Kafka es popular es su excelente ejecución.

3.2.4 Apache Kafka Streaming

Kafka Streams es una librería de procesado de flujos de datos sobre Apache Kafka. Transforman los tópicos de entrada en tópicos de Kafka de salida, le permite hacer esto con código conciso de manera distribuida y tolerante a fallos.
El procesamiento streaming es un paradigma de la programación, equivalente al flujo de datos, procesamiento a eventos y reactivas, que permite a algunas aplicaciones explotar más fácilmente una forma limitada de procesamiento paralelo.

Al estar construida sobre Apache Kafka, nos permite construir aplicaciones con arquitecturas realmente orientadas a eventos, a la vez que aprovechamos sus capacidades subyacentes como la escalabilidad, con órdenes de magnitud en torno al millón de mensajes procesados por segundo, y tolerancia a fallos, a través del particionado de tópicos y su replicación en los miembros del clúster.

3.2.5 Comparación entre Kafka Streaming y Flink

Apache Flink se basa en una arquitectura de clúster (máster y nodos), mas precisamente na arquitectura Kappa. La arquitectura Kappa es una simplificación de la arquitectura Lamba, es decir, un sistema de arquitectura Lamba es el sistema de procesamiento por capa de lotes removido. Para reemplazar el procesamiento por lotes, los datos son introducidos rápidamente por sistema de streaming. Apache Flink es conocido con las características de ejecutar los programas por lotes, secuencias y con alta velocidad, los clústeres siempre están disponibles y implementan de forma independiente permitiendo utilizar un mecanismo de control para garantizar exactamente una vez los resultados en caso de fallas (reprocesamiento a través de los savepoints sin sacrificar la latencia o el rendimiento).

Principales ventajas de Flink

- Capaz de trabajar con sistemas de archivos aparte de los sistemas de archivos de Hadoop (HDFS)
- Procesamiento de datos rápidos
- Modelo de programación unificada (batch y stream)
- Capaz de analizar datos de flujo a una velocidad extrema segura
- Propia biblioteca de Aprendizaje automático (ML), FlinkML para tratar problemas con
- Machine Learning (ML)
- Soporte para consultas iterativas y algoritmos de forma nativa
- Admite muchos operadores nuevos junto con los modelos MapReduce incorporados.
- Soporte de tolerancia a fallos.
- Posee diferentes características de ventanas para los datos transmitidos requeridos por las tecnologías modernas de Big Data (para manejar flujos de datos constantes, • Flink divide los datos que llegan en segmentos seguras la marca de tiempo, el recuento y otros criterios, esto se denomina ventana).

Principales desventajas de Flink

- La ejecución de tuberías (papeline) en Flink tiene limitaciones con respecto a la administración de la memoria (larga ejecución) y la tolerancia a fallas.
- Apache Flink usa bytes brutos como representación de datos internos.
- No tiene API maduras para consultar datos (la API de la tabla de Flink no está del todo en comparación con otros competidores)
- Las API de integración de origen de datos no son las mejores y están limitadas en opciones.

Apache Kafka es un potente motor de procesamiento de stream, incorporando en las aplicaciones de Java estándar para el procesamiento. Las aplicaciones Java son

particularmente adecuadas para construir aplicaciones reactivas, microservicios y sistemas controlados por eventos. La API de Kafka es un componente nativo de Apache Kafka, es una solución de procesamiento stream, altamente escalables, elásticas y de tolerancia a fallas, distribuidos y simples. El objetivo es simplificar el procesamiento stream para que sea posible ser accesible como un modelo de programación para aplicaciones en general. La biblioteca es incorporable sin clúster, solo Kafka y su aplicación, así es posible concentrarse en crear aplicaciones que impulsen su negocio en lugar de crear clústeres, hace que sea más accesible para los desarrolladores de aplicaciones que buscan realizar un procesamiento continuo, ya que se integra con las herramientas empaquetando, implementación, monitoreo y operaciones de la empresa. Es totalmente integrado con las abstracciones del núcleo en Kafka. Los puntos fuertes de Kafka, es la conmutación por error, la elasticidad, la tolerancia a fallos, la escalabilidad y seguridad, permitiendo también integración completa de las abstracciones de las secuencias y de las tablas, que pueden utilizar dentro de la aplicación, para aplicar por ejemplo operaciones de join de alto rendimiento y consultas continuas.

Principales ventajas de Kafka

- Kafka Producer (API Kafka) es simples de usar, es asíncrono y obtiene devolución de llamadas. Es aplicado perfectamente para aplicaciones que emiten flujos de datos de logs, secuencias de clics, IoT, etc.
- Kafka Connect Source (API Kafka) es un marco completo construido sobre la API del productor. Fue creado para que los desarrolladores obtengan una API más agradable para la distribución de tareas del productor para procesamiento paralelo, y un mecanismo fácil para reanudar a sus productores. El producto final es una gran variedad de conectores disponibles que puede aprovechar hoy para incorporar datos de la mayoría de sus fuentes, sin escribir una sola línea de código.
- Kafka Consumer (API Kafka) muy simple, funciona usando Groupos de consumidores para que los tópicos puedan consumir en paralelo.
- Kafka Connect Sink (API Kafka) permite aprovechar el ecosistema de los conectores Kafka para realizar su ETL de transmisión sin escribir línea de código.

- Kafka Stream permite escribir un DSL, leer los datos de Kafka en tiempo real y almacenar nuevamente en el Kafka.
- KSQL no se ejecuta directamente parte de la API de Kafka, envoltura encima de Kafka Streams. Kafka Streams le permite escribir algunas topologías complejas, requiere un conocimiento sustancial de programación y puede ser más difícil de leer, especialmente para los recién llegados. KSQL quiere abstraer esa complejidad al proporcionarle un semántico de SQL (no ANSI)

Principales desventajas de Kafka Stream

- Es firmemente acoplado con Kafka, no puede utilizar sin Kafka.
- Es muy nuevo, todavía hay que ser probado en las compañías.
- No se aplica para trabajos pesados, como Spark Streaming, Flink.

Las diferencias fundamentales entre Flink y API Stream es la formas y manera de implementación y administración (lo que a menudo tiene implicaciones para quién posee estas aplicaciones desde una perspectiva organizativa) y la manera de como se coordina el procesamiento paralelo (tolerancia a fallas).

3.2.6 Ejemplos de Proyectos

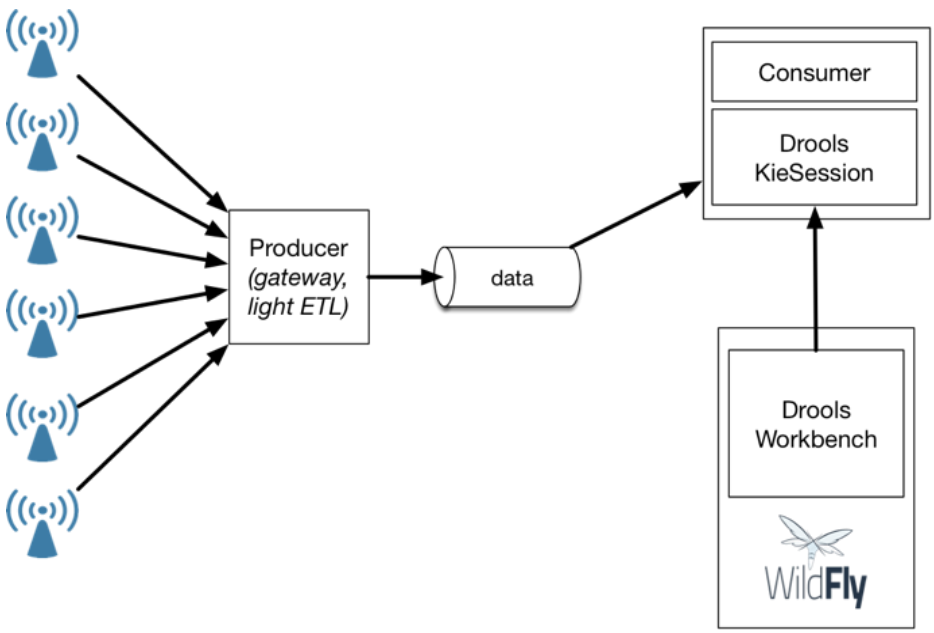
Este apartado tiene la finalidad de analizar proyectos interesantes y parecidos para el caso de estudio. Para el inicio del desarrollo es necesario hacer una investigación de proyectos que se utilizan las herramientas similares para tener una idea base de lo como están utilizando las herramientas y lo que es necesario para mejorar.

3.2.6.1 Análisis de una integración un motor de reglas, KafkaStream y IoT

Según el proyecto realizado por Dumoulin, Mathieu nos explica la arquitectura que se ha utilizado para integrar un motor de reglas con un sistema de Streaming (Kafka) utilizado en casos de IoT. La cantidad de información que generan los sensores es grande, la idea es utilizar dispositivos conectados a una red, fuentes de información como dispositivos como entrada (Ex. como hogares inteligentes, agricultura inteligente, industria 4.0 o datos de telecomunicaciones), siguiendo de un filtraje, aplicación de reglas en los campos de nivel de importancia de acuerdo con las

reglas. La arquitectura utiliza diversas fuentes de inyección de datos (CEP – Complex Event Processing), utiliza procesamiento en tiempo real a través de streaming para toma de acciones de manera inmediata, en lugar del tradicional análisis de datos históricos (batching) que se ejecutan generalmente los procesos por lotes, que suele tener un horario específico de ejecución de la tarea, para no afectar el desempeño de los demás programas que se están ejecutando en paralelo.
Se utiliza el CEP por motivo de su con la agilidad y la obtención de comportamientos complejos de la interacción de múltiples reglas simples que se aplican en tiempo real, en memoria.
La solución propuesta utiliza fuentes de sensores (cajas registrados, logs, etc.) y con un ETL se agregan en el stream. Los datos son consumidos por un programa que envía los datos al Drools KieSession donde el motor de reglas utiliza la coincidencia de patrones para elegir qué reglas pueden activarse en función de los hechos presentes en la memoria. (Ilustración 10)

Ilustración 10 - Representación arquitectura por Dumoulin, Mathie.



Fuente: https://mapr.com/blog/better-complex-event-processing-scale-using-microservices-based-streaming-architecture-part-1/assets/otherpageimages/1917blog/picture2.png

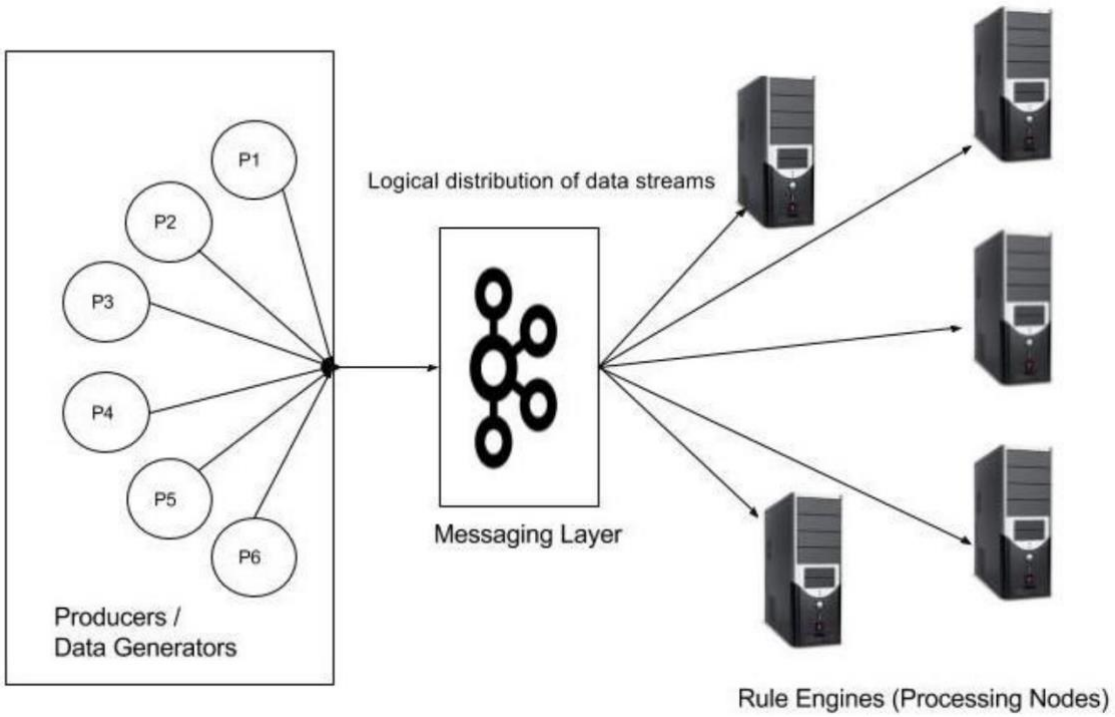
Además, utiliza las reglas del Drools Workbench que proporciona un editor de reglas GUI para utilizar como control de versiones como repositorio para las reglas que se implementen en producción.

3.2.6.2 Análisis de una arquitectura aplicando inferencias paralelas con Kafka

Según la investigación realizado por Murali Krishna, P y Kumar Baruah, Pallav, desarrollarán una arquitectura de prueba de análisis de rendimiento aplicando inferencias lógicas en paralelo, utilizando modelo de reglas y modelo de procesamiento de streaming. Se utiliza datos de transacciones bancarias como datos de entrada como pruebas con el objetivo de mostrar su escalabilidad, rendimiento a través del experimento.

Se utiliza el Kafka la herramienta de intermediación de mensajes para transmitir los datos recibidos (transacciones bancarias) al Apache Drools para aplicar la ejecución de reglas de inferencia lógicamente paralelos. En Kafka es posible distribuir lógicamente los datos definiendo la cantidad de particiones, dividiendo la carga de distribución. Utiliza el Kafka para distribuir lógicamente los datos y el Apache Drools para aplicar el procesamiento de inferencia en cada nodo. (Ilustración 11)

Ilustración 11 - Representación de la arquitectura de aplicación de motor de reglas en Kafka.



El trabajo se concluye que utilizando más particiones para procesamiento de datos en el tiempo de ejecución mejora la performance. Las reglas de inferencia sólo funcionan apenas si el dato está en el mismo nodo de ejecución, este estudio fue desarrollado como la premisa de cada clave esté en cada nodo. Esto implica que las inferencias que está en diferentes claves no es posible juntar las clases para la misma partición.

3.3 Prueba de ejecución con Kafka Streaming

El motivo de utilizar Kafka Streaming como framework para desarrollo es porque trabaja como una API que se adapta a cualquier aplicación Java. La integración con el Drools es más fácil y rápido por el motivo que ambos son nativos con el mismo lenguaje de programación.

El arranque del Kafka es necesario ejecutar el servidor de Zookeeper para administrar los procesos paraleles de Kafka. (Código 1)

```
Kins-MacBook-Pro:kafka_2.12-2.1.1 kintat$ \
> ./bin/zookeeper-server-start.sh \
```

```
> ./config/zookeeper.properties
```

Código 1 - Comando para arrancar del servidor Zookeeper pruebas con el Kafka Streaming.

```
[2019-06-16 14:42:37,657] INFO Server environment:os.name=Mac OS X
(org.apache.zookeeper.server.ZooKeeperServer)
[2019-06-16 14:42:37,657] INFO Server environment:os.arch=x86_64
(org.apache.zookeeper.server.ZooKeeperServer)
[2019-06-16 14:42:37,657] INFO Server environment:os.version=10.14.5
(org.apache.zookeeper.server.ZooKeeperServer)
[2019-06-16 14:42:37,657] INFO Server environment:user.name=kintat
(org.apache.zookeeper.server.ZooKeeperServer)
[2019-06-16 14:42:37,657] INFO Server environment:user.home=/Users/kintat
(org.apache.zookeeper.server.ZooKeeperServer)
[2019-06-16 14:42:37,657] INFO Server
environment:user.dir=/Users/kintat/Documents/kafka_2.12-2.1.1
(org.apache.zookeeper.server.ZooKeeperServer)
[2019-06-16 14:42:37,663] INFO tickTime set to 3000
(org.apache.zookeeper.server.ZooKeeperServer)
[2019-06-16 14:42:37,663] INFO minSessionTimeout set to -1
(org.apache.zookeeper.server.ZooKeeperServer)
[2019-06-16 14:42:37,663] INFO maxSessionTimeout set to -1
(org.apache.zookeeper.server.ZooKeeperServer)
[2019-06-16 14:42:37,679] INFO Using
org.apache.zookeeper.server.NIOServerCnxnFactory as server connection factory
(org.apache.zookeeper.server.ServerCnxnFactory)
[2019-06-16 14:42:37,696] INFO binding to port 0.0.0.0/0.0.0.0:2181
(org.apache.zookeeper.server.NIOServerCnxnFactory)
```

Código 2 - Resultado del arranque del servidor Zookeeper pruebas con el Kafka Streaming.

Una vez que el servidor estar arrancado es posible ejecutar el servidor Kafka.

```
Kins-MacBook-Pro:kafka_2.12-2.1.1 kintat$ \
> ./bin/kafka-server-start.sh \
> ./config/server.properties
```

Código 3 - Comando para arrancar del servidor Kafka pruebas con el Kafka Streaming.

```
expired offsets in 5 milliseconds. (kafka.coordinator.group.GroupMetadataManager)
[2019-06-16 15:04:03,779] INFO [ProducerId Manager 0]: Acquired new producerId block
(brokerId:0,blockStartProducerId:0,blockEndProducerId:999) by writing to Zk with path
version 1 (kafka.coordinator.transaction.ProducerIdManager)
[2019-06-16 15:04:03,801] INFO [TransactionCoordinator id=0] Starting up.
(kafka.coordinator.transaction.TransactionCoordinator)
```

```
[2019-06-16 15:04:03,803] INFO [TransactionCoordinator id=0] Startup complete.
(kafka.coordinator.transaction.TransactionCoordinator)
[2019-06-16 15:04:03,803] INFO [Transaction Marker Channel Manager 0]: Starting
(kafka.coordinator.transaction.TransactionMarkerChannelManager)
[2019-06-16 15:04:03,851] INFO [/config/changes-event-process-thread]: Starting
(kafka.common.ZkNodeChangeNotificationListener$ChangeEventProcessThread)
[2019-06-16 15:04:03,864] INFO [SocketServer brokerId=0] Started processors for 1
acceptors (kafka.network.SocketServer)
[2019-06-16 15:04:03,868] INFO Kafka version : 2.1.1
(org.apache.kafka.common.utils.AppInfoParser)
[2019-06-16 15:04:03,868] INFO Kafka commitId : 21234bee31165527
(org.apache.kafka.common.utils.AppInfoParser)
[2019-06-16 15:04:03,869] INFO [KafkaServer id=0] started (kafka.server.KafkaServer)
```

Código 4 - Resultado del arranque del servidor Kafka pruebas con el Kafka Streaming.

3.3.1 Contador de palabras con Kafka Stream

Ejecución de ejemplo el WordCount del Kafka para demonstrar la diferencia entre Flink. Primero es necesario crear un tópico para que sea el local de almacenamiento de los mensajes de entrada datos. (Código 5).

```
Kins-MacBook-Pro:bin kintat$ ./kafka-topics.sh \
> --create                              \
> --zookeeper localhost:2181                \
> --replication-factor 1                    \
> --partitions 1                            \
> --topic streams-plaintext-input
Created topic "streams-plaintext-input".
Kins-MacBook-Pro:bin kintat$
```

Código 5 - Creación de un tópico llamado streams-plaintext-input.

Para que sea posible procesar el contador de palabras, es necesario observar los datos que constantemente llegan en el tópico creado anteriormente. El programa WordCountDemo es un ejemplo propio del Kafka, tiene configurado y compilado para escuchar el tópico “streams-plaintext-input” como entrada de datos. Al ejecutar este programa automáticamente es lanzada un proceso que vigila el tópico. El momento que sufre alguna modificación o producción de un nuevo dato, un disparo es enviado al programa de procesamiento (WordCountDemo) para efectuar tu función. Podemos observar de acuerdo con el Código 6.

```
Kins-MacBook-Pro:bin kintat$ ./kafka-run-class.sh \
> org.apache.kafka.streams.examples.wordcount.WordCountDemo  \
>
[2019-06-16 15:37:45,825] WARN The configuration 'admin.retries' was supplied but isn't a known config. (org.apache.kafka.clients.consumer.ConsumerConfig)
[2019-06-16 15:37:45,826] WARN The configuration 'admin.retry.backoff.ms' was supplied but isn't a known config. (org.apache.kafka.clients.consumer.ConsumerConfig)
```

Código 6 - Ejecución del ejemplo de WordCount propio del Kafka.

La inserción de los datos utilizase el producer (productor) es un script[17] que el Kafka nos proporciona para ejecutar por líneas de código en el terminal. Al ejecutar el producer es necesario informar el tópico que será enviado los registros de entrada, para luego ser procesado por el contador de palabras (Código 6). Una vez ejecutado, abre una conexión con el tópico que nos permite el envío de datos, conforme el Código 7.

```
Kins-MacBook-Pro:bin kintat$ ./kafka-console-producer.sh     \
> --broker-list localhost:9092           \
> --topic streams-plaintext-input        \
>
>tan kin tat
>contando palabras
>prueba 1 2 3
```

[17] https://bioinf.comav.upv.es/courses/unix/scripts_bash.html

```
>
```

Código 7 - Ejecución del producer enviando datos de entrada.

De acuerdo con los registros de entrada, el programa WordCountDemo (contador de palabras) es ejecutado. Para verificar los resultados del procesamiento es necesario utilizar el consumer (consumidor) es que un script que el Kafka nos proporciona para ejecutar en el terminal. De acuerdo con Código 8 podemos observar que conforme los datos se van siendo añadidos en el tópico, automáticamente el procesamiento es activado.

```
Kins-MacBook-Pro:bin kintat$ ./kafka-console-consumer.sh \
> --bootstrap-server localhost:9092                    \
> --topic streams-wordcount-output                     \
> --from-beginning                                     \
> --formatter kafka.tools.DefaultMessageFormatter      \
> --property print.key=true                            \
> --property key.deserializer=org.apache.kafka.common.serialization.StringDeserializer \
> --property value.deserializer=org.apache.kafka.common.serialization.LongDeserializer \
>
tan     1
kin     1
tat     1
contando    1
palabras    1
prueba      1
1       1
2       1
3       1
```

Código 8 - Mostrando el contador de palabras de salida.

3.3.2 Prueba de ejecución con Flink

La instalación del Flink en el macOS es efecutado por el HomeBrew[18] (gestor de paquete para macOS), los programas se instalan en el directorio conforme el Código 9.

```
Kins-MacBook-Pro:~ kintat$ cd /usr/local/Cellar/apache-flink/1.8.0/
Kins-MacBook-Pro:1.8.0 kintat$
```

Código 9 - Directorio no cual se encuentra el Flink instalado por el Homebrew.

Flink trabaja en una arquitectura de clúster, para empezar, es necesario ejecutar el clúster de Flink para poder arrancar los Task Managers[19] y Slots[20] para que sea posible ejecutar sus tareas. El Código 10 representa la ejecución de inicio del clúster.

```
Kins-MacBook-Pro:1.8.0 kintat$ ./libexec/bin/start-cluster.sh
Starting cluster.
Starting standalonesession daemon on host Kins-MacBook-Pro.local.
Starting taskexecutor daemon on host Kins-MacBook-Pro.local.
Kins-MacBook-Pro:1.8.0 kintat$
```

Código 10 - Inicia el clúster del Apache Flink.

Para asegurar el arranque del Flink y verificar si los Task Managers, Task Slots están activados, la interfaz de control de ejecución de Kafka (http://localhost:8081), debe estar igual de acuerdo con la Ilustración 12. Es la página frontal que se utiliza múltiples tareas: controlar los Jobs en ejecución, observar la información de las ejecuciones como también ejecutar las tareas desde la interfaz.

[18] https://brew.sh/index_es

[19] El TaskManager es responsable de ejecutar las tareas individuales de un trabajo de Flink.

[20] Recursos de ejecución del Flink

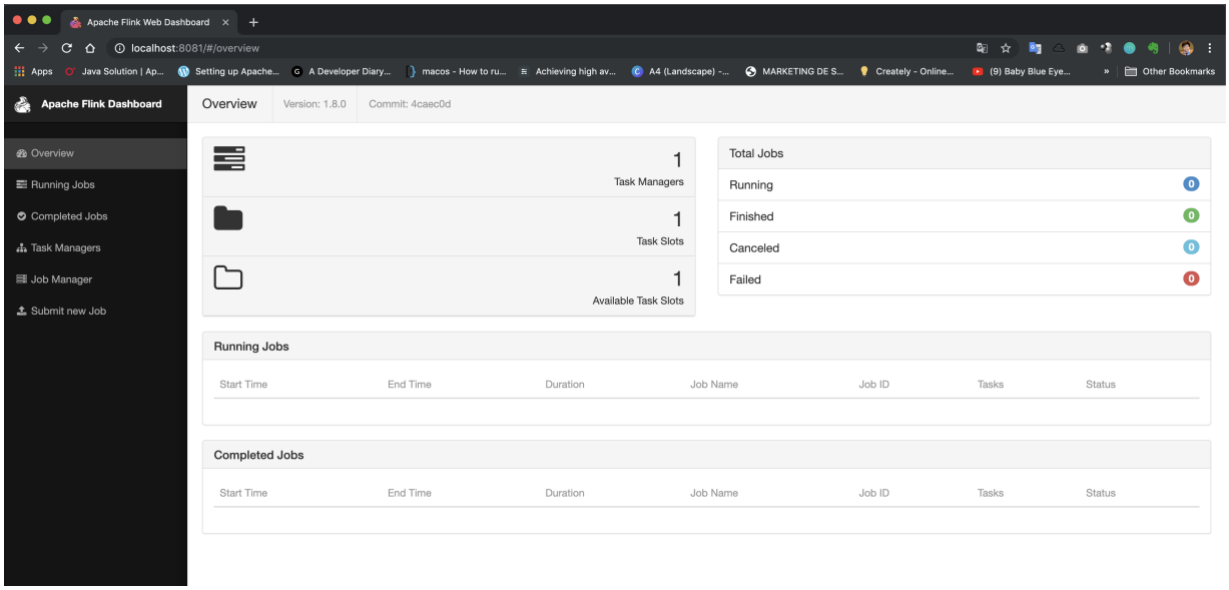


Ilustración 12 - Interfaz web de las ejecuciones del Flink.

Para enviar los textos de entrada es necesario arrancar un servidor Netcat conectando en una puerta para que sea posible el envío de los datos de entrada. Por ejemplo, utilizamos la puerta 9000 para enviar los datos. (Código 11).

```
Kins-MacBook-Pro:~ kintat$ nc -l 9000


tan kin tat
contando cantidad de palabras
tan kin tat
prueba 1 2 3
```

Código 11 - Iniciar un servidor conectando en la puerta 9000 para envío los datos de entrada.

El WordCount (contador de palabras) de ejemplo propio de Flink tiene como parámetro de entrada el puerto (Código 12). Semejante con el Kafka (Código 6), vigila los datos de entrada en el puerto informado, para cuando algún registro de entrada es enviado se ejecuta el procesamiento.

```
Kins-MacBook-Pro:1.8.0 kintat$ ./libexec/bin/flink run
./libexec/examples/streaming/WordCount.jar --port 9000
Starting execution of program
```

```
Executing WordCount example with default input data set.
Use --input to specify file input.
Printing result to stdout. Use --output to specify output path.
Program execution finished
Job with JobID 413c15179ad3b418655b8b8ef365ca5d has finished.
Job Runtime: 72 ms
Kins-MacBook-Pro:1.8.0 kintat$
```

Código 12 - Ejecución del contador de palabras consumiendo los datos de entrada de la puerta 9000.

Una vez procesado, para ver el resultado del procesamiento diferente del Kafka que utiliza el consumer para consulta, el Flink resultado en los logs de ejecución. (Código 13).

```
Kins-MacBook-Pro:1.8.0 kintat$ tail -f libexec/log/flink-*-taskexecutor-*.out

==> libexec/log/flink-kintat-taskexecutor-2-Kins-MacBook-Pro.local.out <==

==> libexec/log/flink-kintat-taskexecutor-0-Kins-MacBook-Pro.local.out <==
tan : 1
tat : 1
kin : 1
contando : 1
tat : 1
kin : 1
tan : 1
palabras : 1
de : 1
cantidad : 1
prueba : 1
3 : 1
2 : 1
```

Código 13 - Resultado del contador de palabras por Apache Flink.

3.4 Propuesta Técnica

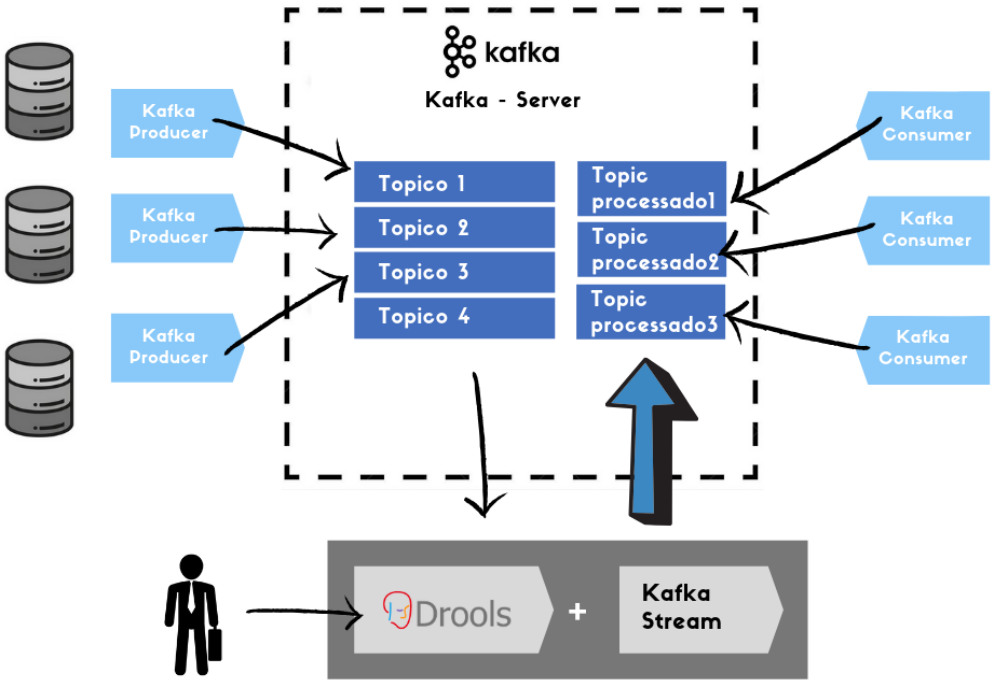
La arquitectura (Ilustración 13) es dividido en tres capas: Productores (datos de entrada), Procesamiento (Kafka Stream y Drools), Consumidores (Resultados). El responsable de intermediación y envío de mensajes es el Kafka, almacena los datos de entrada enviados por los productores y distribuye los datos para los consumidores de acuerdo con las solicitaciones.

Los productores se encargan de enviar los datos, almacenando en los tópicos de entrada, generando un mensaje de acuerdo sobre el tópico específico.

El procesamiento se base en la integración del Kafka Stream con el Drools, el motivo es que el Kafka Stream permite hacer le procesamiento en tiempo real pudiendo hacer transformaciones stateless (filter, map, transform, etc) y stateful (contador, agregations, joins, etc). Drools se utiliza para definir las reglas de negocio, en la facilidad y control de las reglas de negocio, simplificando los proyectos y reglas de alto complejidad.

Los consumidores son la consulta y ejecución de servicios del resultado, son los valores del procesado como: resultado de una investigación de clientes, identificación de algún problema, etc.

Ilustración 13 - Arquitectura integración Kafka Stream aplicación Drools con regla de negocio.



3.4.1 Datos de Entrada

El conjunto de datos original nos proporciona los siguientes tipos de datos:

Datos bancarios del cliente

- Edad – La edad del cliente.
- Trabajo – Tipo de trabajo (desconocido, administrador, desempleado, dirección, empleada doméstica, empresario, estudiante, obrero, trabajador por cuenta propia, jubilado, técnico, servicios).
- Matrimonial – Estado civil (casado, divorciado, soltero)
- Educación – Nivel de estudio (desconocido, analfabeto, básico 4, básico 6, básico 9, escuela secundaria, curso profesional, universidad)
- Defecto – Posee crédito por defecto. (si, no)
- Saldo – Saldo medio anual, en euros.
- Vivienda – Préstamo de vivienda (si, no)
- Préstamo – Préstamo personal (si, no)

Datos relacionados con el último contacto con el cliente

- Contacto – Tipo de comunicación de contacto (desconocido, teléfono, celular)
- Días – Último día de contacto del mes.
- Meses – Último mes de contacto del año. (jan, feb, mar, abr, may, jun, jul, ago, sep, oct, nov, dec).
- Duración – Duración del ultimo contacto (en segundos)
- Campaña – Número de contactos realizados durante la campaña.
- Días – Número de días que pasaron después de que el cliente fue contactado por última vez desde una campaña anterior.
- Anterior – Cantidad de veces contactado antes de esta campaña y para este cliente.
- Poutcome – Resultado de campaña de marketing anterior (desconocido, otro, fracaso, exito)

Datos contexto social económico

Emp.var.rate – Tasa de variación del empleo.
Cons.price.idx – Índice de precios al consumidor.
Cons.conf.idx – Índice de confianza del consumidor
Euribor3m – (tipo de interés aplicado a las operaciones entre bancos de Europa) tasa de 3 meses.
Num. Empleados – Número de empleados.

Realiza se una transformación de los datos originales para un conjunto de datos de envíos streaming. Los datos de streaming es necesario informar el valor de la clave porque será el factor fundamental para identificar el respectivo cliente después del procesado. Es una ventaja porque es posible hacer junciones con el ID único para conectar diversas fuentes de información.

Tabla 1 - Transformación de los datos de entrada.

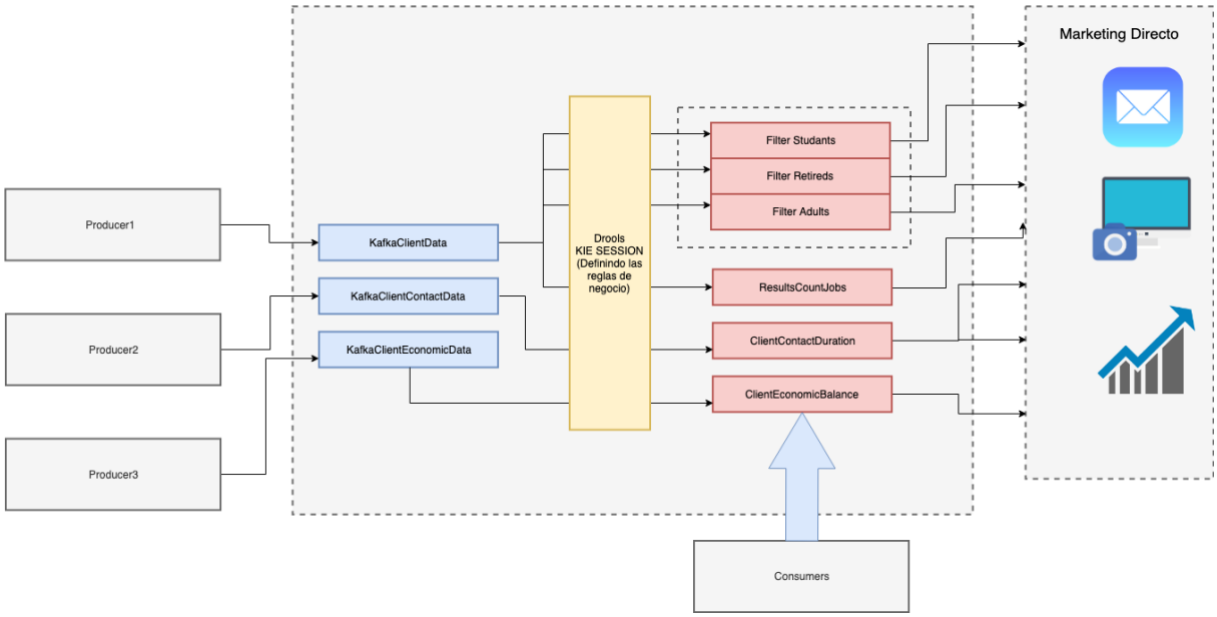
Datos bancarios del cliente	Datos último contacto con el cliente	Datos contexto social económico
Id	**Id**	**Id**
Edad	Contacto	Emp.var.rate
Trabajo	Día	Cons.price.idx
Matrimonial	Meses	Cons.conf.idx
Educación	**Dia de la semana (mon, tue. wed…)**	Euribor3m
Defecto	Duración	Num. Empleados
~~Saldo~~	Campaña	**Saldo**
Vivienda	Días	
Préstamo	Anterior	
	Poutcome	

Para la transformación de los datos en streaming, es necesario tener en cuenta que enviados los datos en un flujo continuo. Por lo tanto, fue definido el ID único del cliente para los 3 (tres) tipos de fuentes de entrada (ID cliente). Modificación del saldo en los datos de cliente para Datos de contexto social y pues en un entorno real el saldo cambia a todo instante. Añadir él día de la semana como nueva una información (mon, tue, wed…), por refinar más las informaciones de los clientes.

3.4.2 Definición de las reglas de negocio

De acuerdo con Ilustración 14, fue definido algunas reglas de negocio para que sea aplicado en los datos de entrada con la herramienta Drools. Los datos de entrada se almacenan en los tópicos del Kafka, y luego serán procesados por Kafka Stream ejecutando las reglas de negocio. Si validan las reglas será almacenado en tópicos de salida en Kafka como resultado. La idea es como identificación de grupos de clientes que necesitan de inspección, promoción de un nuevo producto, etc.

Ilustración 14 - Arquitectura definiendo las reglas de negocio de la aplicación



- **Perfil Estudiantil:** El perfil estudiantil, son los jóvenes entre 18-25 años que están en la universidad, son los futuros clientes que empiezan la carrera de trabajo. El objetivo es encontrar clientes para recomendar tipos de eventos relaciones al publico universitario, como de una nueva tarjeta creada con los beneficios a ayuda a los universitarios en la parte de gestión con la institución, de o un nuevo producto.
- **Perfil Jubilados:** El perfil jubilado, es encontrar los clientes mayores de 50 años, por ejemplo, para recomendar un tipo de cuenta especial, añadir promociones de empresas terceras de salud, alimentación, transporte asociados para facilitar con el cliente.
- **Perfil Adultos:** El perfil adulto, seria perfiles mayoría del público, entre 25-45 años, por ejemplo, que no tengan casa y sin deuda, recomendando por ejemplo promociones de casa propia a alguna promoción similar.
- **Cantidad total de funcionarios por tipo de trabajo:** La idea es identificar la cantidad de funcionarios divididos por tipo de trabajo, es para tener una idea del cual es la proporción de empleo de los clientes, un poder crear nuevos productos atendiendo mejor los clientes.
- **Contacto en los clientes mayores 3 meses desde el último contacto:** Es identificar los clientes que fueron contactos mayores de 3 (tres) meses

de intervalo de tiempo, para volver a contactar, o enviar un email notificando si un productor similar de el último contacto.

- **Cantidad total de duración de tiempo de la llamada:** Suma total de la cantidad de tiempo que el cliente estuvo en la llamada, puede ser por motivo de reclamación o por motivo interés del cliente, es la identificar de los clientes para volver a contactar para resolver el problema o tener más aproximación con el cliente.

3.4.3 Drools reglas de negocio (.drl)

La Código 14 y Código 15 son los filtros definido con el lenguaje de reglas para aplicar en el Kafka Stream. Percebe se que es un lenguaje fácil de entender, es un lenguaje a nivel de negocio que se organiza en lógica de condición, identificando en que acción que se debe aplicar y realizar.

Se define los filtros para buscar los perfiles estudiantiles, jubilados, adultos en los datos de entrada de cliente (Código 14).

```
// Perfil Estudantil
// Jovenes mayores de 18-25 años que están en la universidad que no hay ninguna deuda.
rule "Filter Perfil Estudantil"
   when
      b : Client(   (getAge() > 18 && getAge() < 25)
                                          && (getJob() == "student")
                                          && (getEducation() == "university.degree")
                                          && (getDebtor() == "no")
                     )
   then
         KafkaRuleClientProducer.addRuleDroolsClient("KafkaPerfilEstudantil",
Constants.topicFilterClientEstudant, b);
end


// Perfil Jubilados
// Jubilados mayores que 50 años que están solteros que no tienen casa y que están con deudas.
rule "Perfil Aposentados"
```

```
  when
    b : Client(   (getAge() > 50)
                && (getMarital() == "single")
                && (getHousing() == "no")
                && (getDebtor() == "yes")
               )
  then
        KafkaRuleClientProducer.addRuleDroolsClient("KafkaPerfilEstudantil",
Constants.topicFilterClientRetired, b);

end

// Perfil Adultos
// Casados entre 25-45 que no tengan casa sin deuda.
rule "Perfil Adultos"
  when
    b : Client(   (getAge() > 25 &&  getAge() < 45)
                                  && (getDebtor() == "no")
              )
  then
        KafkaRuleClientProducer.addRuleDroolsClient("KafkaPerfilEstudantil",
Constants.topicFilterClientAdults, b);

                                                  end
```

Código 14 - Definición de las reglas de negocio (perfiles)

Se aplica la regla para encontrar el cliente con mayores de 3 meses desde el último contacto. (Código 15).

```
// Contacto Clientes
// Clientes contactados mayores que 3 meses. Para volver a contactar los clientes.
rule "Client Contact"
  when
    b : ClientContact(   (Utils.transformMonthToInt(getMonth()) < (new Date().getMonth())-3)
```

```
                                        || (Utils.transformMonthToInt(getMonth()) == (new
Date().getMonth())-3
                                        && (getDay() <= new Date().getDate() ))
                                       )
  then
        System.out.println("Entrei aqui");
        KafkaRuleClientProducer.addRuleDroolsClientContact("KafkaContact",
Constants.topicClientContactBackContact, b);
end
```

Código 15 - Definición de la regla para contacto del cliente.

3.4.4 Aplicación de reglas por tabla de decisión (.xls)

Por motivo de facilitar el sector de negocios, es posible aplicar las reglas por tablas de decisión, realiza el mismo proceso, pero su utilización es para abstraer más la parte técnica para que apenas concentre en los cambios de negocios. Es utilizando en los escenarios donde el cliente no tiene poco conocimiento de lenguaje de programación. En el aparcado de anexos se encuentra la tabla de decisión realizada como alternativa para definición de las reglas de negocio.

Los expertos del negocio, equipo médico y gestores del sistema, por ejemplo, poseen una cultura de edición de reglas y son capaces de gestionar el control de historias, la edición de reglas en un archivo de tablas de Excel agiliza su gestión evitando problemas de sintaxis y reglas de programación.

3.4.5 Resultados

Arranque del servidor Zookeeper

Arranque del servidor Zookeeper, tiene la función de centralizar las diversas tareas por mantenimiento de configuración, naming, sincronización distribuida o servicios de agrupación. El proceso de ejecutar el Zookeeper es el mismo utilizado él ejemplo del contador de palabras (Código 1**Error! Reference source not found.**).

Arranque del servidor Kafka

Para iniciar el servidor Kafka, es necesario estar activo el servidor Zookeeper. Es necesario el arranque del servidor para ejecutar las tareas y servicios del Kafka. (Código 3, Código 4).

Entrada de datos

Una vez que los servidores de Zookeeper y Kafka están activos, es posible enviar los datos de entrada a los tópicos de Kafka.

Por motivos de prueba de concepto, fue hecho una selección de los registros de entrada, para que sea posible probar y tener identificar el funcionamiento de las reglas de negocio. La Ilustración 15 representa los registros de entrada para el tópico de datos de clientes.

Ilustración 15 - Datos de clientes seleccionados como entrada del tópico cliente.

9	56	housemaid	married	basic.4y	no	no	no
2	57	services	married	high.school	yes	no	no
8	37	services	married	high.school	no	yes	no
9	73	retired	divorced	basic.4y	no	no	yes
3	56	services	married	high.school	no	no	yes
5	45	services	married	basic.9y	yes	no	no
5	59	admin.	married	professional.	no	no	no
2	41	blue-collar	married	unknown	yes	no	no
4	23	student	single	university.de	no	yes	no
6	25	services	single	high.school	no	yes	no
4	30	blue-collar	married	unknown	yes	no	no
6	25	services	single	high.school	no	yes	no
9	29	blue-collar	single	high.school	no	no	yes
3	25	student	single	university.de	no	yes	no
5	35	blue-collar	married	basic.6y	no	yes	no
8	54	retired	married	basic.9y	yes	yes	yes
5	28	blue-collar	married	basic.6y	no	yes	no
1	73	retired	married	basic.4y	no	yes	no
6	21	student	single	university.de	no	yes	no
2	39	managemen	single	basic.9y	yes	no	no
7	30	unemployed	married	high.school	no	no	no
5	55	blue-collar	married	basic.4y	yes	yes	no
1	55	retired	single	high.school	no	yes	no
8	41	technician	single	high.school	no	yes	no
10	37	admin.	married	high.school	no	yes	no
6	35	technician	married	university.de	no	no	yes
5	59	technician	married	unknown	no	yes	no
5	39	self-employe	married	basic.9y	yes	no	no
6	54	technician	single	university.de	yes	no	no
5	32	unknown	married	university.de	yes	unknown	unknown
6	46	admin.	married	unknown	no	no	no
7	23	student	single	university.de	no	yes	no
7	30	blue-collar	married	university.de	no	no	no
10	54	managemen	married	basic.4y	yes	yes	no
8	54	blue-collar	divorced	basic.4y	no	no	no
8	21	student	single	university.de	no	yes	no
5	34	services	married	high.school	no	no	no
4	52	technician	married	basic.9y	no	yes	no
10	41	admin.	married	university.de	no	yes	no
6	56	technician	married	basic.4y	no	yes	no

El Kafka nos proporciona una API que es posible ejecutar sus servicios a través de lenguaje de programación. La simulación consiste en leer cada registro de entrada del fichero de clientes (.xls) y producir enviando los datos en los tópicos del Kafka. Con ayuda del objeto Producer de Kafka es posible enviar los datos de manera continua, simulando el almacenamiento en tiempo real.

Para verificar si los datos fueron persistidos correctamente en el tópico, utilizamos el Consumer consultar que lo ha almacenado de acuerdo con los datos de entrada. (Código 16 , Código 17)

```
Kins-MacBook-Pro:bin kintat$ ./kafka-console-consumer.sh \
> --topic KafkaClientDataTest1          \
> --from-beginning                      \
> --bootstrap-server localhost:9082     \
>
```

Código 16 - Comando de verificación del tópico datos clientes.

```
egree","debtor":"yes","housing":"no","loan":"no"}
{"id":5,"age":32,"job":"unknown","marital":"married","education":"university.degree","debtor":"
yes","housing":"unknown","loan":"unknown"}
{"id":6,"age":46,"job":"admin.","marital":"married","education":"unknown","debtor":"no","housin
g":"no","loan":"no"}
{"id":7,"age":23,"job":"student","marital":"single","education":"university.degree","debtor":"no",
"housing":"yes","loan":"no"}
{"id":7,"age":30,"job":"blue-
collar","marital":"married","education":"university.degree","debtor":"no","housing":"no","loan":"
no"}
{"id":10,"age":54,"job":"management","marital":"married","education":"basic.4y","debtor":"yes"
,"housing":"yes","loan":"no"}
{"id":8,"age":54,"job":"blue-
collar","marital":"divorced","education":"basic.4y","debtor":"no","housing":"no","loan":"no"}
{"id":8,"age":21,"job":"student","marital":"single","education":"university.degree","debtor":"no",
"housing":"yes","loan":"no"}
{"id":5,"age":34,"job":"services","marital":"married","education":"high.school","debtor":"no","ho
using":"no","loan":"no"}
{"id":4,"age":52,"job":"technician","marital":"married","education":"basic.9y","debtor":"no","hou
sing":"yes","loan":"no"}
{"id":10,"age":41,"job":"admin.","marital":"married","education":"university.degree","debtor":"n
o","housing":"yes","loan":"no"}
```

{"id":6,"age":56,"job":"technician","marital":"married","education":"basic.4y","debtor":"no","housing":"yes","loan":"no"}

Código 17 - Resultado de envío de los registros datos clientes.

La Ilustración 16 representa los datos de entrada de cliente contacto, los contactos realizados con él cliente.

Ilustración 16 - Datos de entrada para Client Contacto.

41156	telephone	0	nov	mon	152	5	999	0	nonexistent
41157	cellular	8	nov	mon	545	2	999	0	nonexistent
41158	cellular	11	nov	mon	159	4	999	0	nonexistent
41159	cellular	2	nov	tue	363	1	999	0	nonexistent
41160	cellular	4	nov	tue	514	1	9	4	success
41161	cellular	14	nov	tue	843	1	999	0	nonexistent
41162	cellular	5	nov	tue	510	1	999	1	failure
41163	cellular	20	nov	tue	347	2	4	1	success
41164	cellular	11	nov	tue	385	3	4	2	success
41165	cellular	1	nov	tue	1868	2	10	1	success
41166	cellular	16	nov	wed	403	2	999	0	nonexistent
41167	telephone	1	nov	wed	651	1	999	1	failure
41168	cellular	9	nov	wed	236	3	999	0	nonexistent
41169	cellular	10	nov	wed	144	2	999	0	nonexistent
41170	cellular	22	nov	wed	154	5	999	0	nonexistent
41171	cellular	6	nov	wed	293	2	999	4	failure
41172	telephone	28	nov	thu	112	1	999	0	nonexistent
41173	cellular	10	nov	thu	353	1	999	0	nonexistent
41174	cellular	4	nov	thu	329	1	999	2	failure
41175	cellular	5	nov	thu	208	1	1	6	success
41176	cellular	5	nov	thu	180	1	999	2	failure
41177	cellular	2	nov	thu	360	1	999	0	nonexistent
41178	cellular	4	nov	thu	124	6	999	0	nonexistent
41179	cellular	2	nov	thu	483	2	6	3	success
41180	cellular	19	nov	fri	151	3	999	0	nonexistent
41181	cellular	10	nov	fri	254	2	999	0	nonexistent
41182	cellular	8	nov	fri	281	1	999	0	nonexistent
41183	cellular	1	nov	fri	112	1	9	1	success
41184	cellular	19	nov	fri	334	1	999	0	nonexistent
41185	cellular	17	nov	fri	383	1	999	0	nonexistent
41186	cellular	4	nov	fri	189	2	999	0	nonexistent
41187	cellular	5	nov	fri	442	1	999	0	nonexistent
41188	cellular	28	nov	fri	239	3	999	1	failure

Se ejecuta el Producer correspondiente a los datos de clientes contacto de acuerdo con el Excel (.xls), enviando los datos para el tópico correspondiente a los tópicos de contactos (Código 18, Código 19).

```
Kins-MacBook-Pro:bin kintat$ ./kafka-console-consumer.sh  \
> --topic KafkaClientContactData1     \
> --from-beginning                    \
> --bootstrap-server localhost:9092   \
                                                    >
```

Código 18 - Comando de verificación del tópico datos clientes contacto.


```
n":180,"campaign":2,"pdays":999,"previous":0,"outcome":"nonexistent"}
{"id":92,"contact":"telephone","day":1,"month":"may","dayOfWeek":"mon","duration":48,"camp
aign":1,"pdays":999,"previous":0,"outcome":"nonexistent"}
{"id":93,"contact":"telephone","day":11,"month":"may","dayOfWeek":"mon","duration":213,"ca
mpaign":2,"pdays":999,"previous":0,"outcome":"nonexistent"}
{"id":94,"contact":"telephone","day":8,"month":"may","dayOfWeek":"mon","duration":545,"cam
paign":1,"pdays":999,"previous":0,"outcome":"nonexistent"}
{"id":95,"contact":"telephone","day":17,"month":"may","dayOfWeek":"mon","duration":583,"ca
mpaign":1,"pdays":999,"previous":0,"outcome":"nonexistent"}
{"id":96,"contact":"telephone","day":24,"month":"may","dayOfWeek":"mon","duration":221,"ca
mpaign":1,"pdays":999,"previous":0,"outcome":"nonexistent"}
{"id":97,"contact":"telephone","day":23,"month":"may","dayOfWeek":"mon","duration":426,"ca
mpaign":1,"pdays":999,"previous":0,"outcome":"nonexistent"}
{"id":98,"contact":"telephone","day":25,"month":"may","dayOfWeek":"mon","duration":287,"ca
mpaign":1,"pdays":999,"previous":0,"outcome":"nonexistent"}
{"id":99,"contact":"telephone","day":9,"month":"may","dayOfWeek":"mon","duration":197,"cam
paign":1,"pdays":999,"previous":0,"outcome":"nonexistent"}
{"id":100,"contact":"telephone","day":23,"month":"may","dayOfWeek":"mon","duration":257,"c
ampaign":1,"pdays":999,"previous":0,"outcome":"nonexistent"}
{"id":101,"contact":"telephone","day":18,"month":"may","dayOfWeek":"mon","duration":229,"c
ampaign":1,"pdays":999,"previous":0,"outcome":"nonexistent"}
{"id":102,"contact":"telephone","day":19,"month":"may","dayOfWeek":"mon","duration":55,"ca
mpaign":3,"pdays":999,"previous":0,"outcome":"nonexistent"}
```

Código 19 - Resultado de envío de los registros datos clientes contacto.

La Ilustración 17 representa los datos de entrada de cliente económicos, el estado del saldo correspondiente del cliente, etc.

Ilustración 17 - Datos de entrada para Cliente Económico.

41156	-1.1	94.767	-50.8	1.039	4963.6	256
41157	-1.1	94.767	-50.8	1.039	4963.6	6320
41158	-1.1	94.767	-50.8	1.039	4963.6	1458
41159	-1.1	94.767	-50.8	1.035	4963.6	0
41160	-1.1	94.767	-50.8	1.035	4963.6	132
41161	-1.1	94.767	-50.8	1.035	4963.6	1612
41162	-1.1	94.767	-50.8	1.035	4963.6	4111
41163	-1.1	94.767	-50.8	1.035	4963.6	553
41164	-1.1	94.767	-50.8	1.035	4963.6	3704
41165	-1.1	94.767	-50.8	1.035	4963.6	2658
41166	-1.1	94.767	-50.8	1.03	4963.6	3411
41167	-1.1	94.767	-50.8	1.03	4963.6	407
41168	-1.1	94.767	-50.8	1.03	4963.6	0
41169	-1.1	94.767	-50.8	1.03	4963.6	1
41170	-1.1	94.767	-50.8	1.03	4963.6	10905
41171	-1.1	94.767	-50.8	1.03	4963.6	976
41172	-1.1	94.767	-50.8	1.031	4963.6	0
41173	-1.1	94.767	-50.8	1.031	4963.6	2531
41174	-1.1	94.767	-50.8	1.031	4963.6	1270
41175	-1.1	94.767	-50.8	1.031	4963.6	5359
41176	-1.1	94.767	-50.8	1.031	4963.6	139
41177	-1.1	94.767	-50.8	1.031	4963.6	10596
41178	-1.1	94.767	-50.8	1.031	4963.6	5871
41179	-1.1	94.767	-50.8	1.031	4963.6	80
41180	-1.1	94.767	-50.8	1.028	4963.6	1
41181	-1.1	94.767	-50.8	1.028	4963.6	691
41182	-1.1	94.767	-50.8	1.028	4963.6	4859
41183	-1.1	94.767	-50.8	1.028	4963.6	67
41184	-1.1	94.767	-50.8	1.028	4963.6	10253
41185	-1.1	94.767	-50.8	1.028	4963.6	347
41186	-1.1	94.767	-50.8	1.028	4963.6	9962
41187	-1.1	94.767	-50.8	1.028	4963.6	2239
41188	-1.1	94.767	-50.8	1.028	4963.6	1978

Ejecuta el Producer correspondiente de los registros económico de los clientes, para enviar los registros económicos en el tópico económico. (Código 20, Código 21)

```
Kins-MacBook-Pro:bin kintat$ ./kafka-console-consumer.sh   \
> --topic KafkaClientEconomicData1          \
> --from-beginning                          \
> --bootstrap-server localhost:9092         \
```

```
>
```

Código 20 - Comando de verificación del tópico datos clientes económico.

```
Employed":5191.0,"balance":2586}
{"id":42,"empVarRate":1.1,"consPriceIdx":93994.0,"consConfIdx":-
36.4,"euribor3m":4857.0,"nrEmployed":5191.0,"balance":49}
{"id":43,"empVarRate":1.1,"consPriceIdx":93994.0,"consConfIdx":-
36.4,"euribor3m":4857.0,"nrEmployed":5191.0,"balance":104}
{"id":44,"empVarRate":1.1,"consPriceIdx":93994.0,"consConfIdx":-
36.4,"euribor3m":4857.0,"nrEmployed":5191.0,"balance":529}
{"id":45,"empVarRate":1.1,"consPriceIdx":93994.0,"consConfIdx":-
36.4,"euribor3m":4857.0,"nrEmployed":5191.0,"balance":96}
{"id":46,"empVarRate":1.1,"consPriceIdx":93994.0,"consConfIdx":-
36.4,"euribor3m":4857.0,"nrEmployed":5191.0,"balance":-171}
{"id":47,"empVarRate":1.1,"consPriceIdx":93994.0,"consConfIdx":-
36.4,"euribor3m":4857.0,"nrEmployed":5191.0,"balance":-364}
{"id":48,"empVarRate":1.1,"consPriceIdx":93994.0,"consConfIdx":-
36.4,"euribor3m":4857.0,"nrEmployed":5191.0,"balance":0}
{"id":49,"empVarRate":1.1,"consPriceIdx":93994.0,"consConfIdx":-
36.4,"euribor3m":4857.0,"nrEmployed":5191.0,"balance":0}
{"id":50,"empVarRate":1.1,"consPriceIdx":93994.0,"consConfIdx":-
36.4,"euribor3m":4857.0,"nrEmployed":5191.0,"balance":0}
{"id":51,"empVarRate":1.1,"consPriceIdx":93994.0,"consConfIdx":-
36.4,"euribor3m":4857.0,"nrEmployed":5191.0,"balance":1291}
{"id":52,"empVarRate":1.1,"consPriceIdx":93994.0,"consConfIdx":-
36.4,"euribor3m":4857.0,"nrEmployed":5191.0,"balance":-244}
```

Código 21 - Resultado de envío de los registros datos clientes económicos.

3.4.6 Procesamiento

El procesamiento aplica las reglas de negocio en los datos de entradas en cada tópico de entrada. Para cada registro de entrada, es aplicado las reglas de negocio

definidas anteriormente en tiempo real. Los procesos arrancan de manera asíncrona esperando que algún tópico sufra modificación punto para aplicar las reglas.

Al desarrollar llevase en cuenta el concepto de tipos de datos de Serialización y Deserialización. La serialización un mecanismo para convertir el estado de un objeto en un stream de bytes, la deserialización es el proceso inverso en el que se utiliza el flujo de bytes para recrear el objeto real en la memoria.

El envío de los datos entre los productores, procesamiento y consumidores es de acuerdo con el tipo de Serialización/Deserialización definidos, por que será el tipo de objeto que deberá ser utilizado para consulta y envío.

El Kafka nos proporciona la Stream DSL (Domain Specific Language – Lenguaje específica de dominio). Abstrai funciones internas para stream y tables na forma de KStream, KTable y GlobalKTable. Es un tipo de programación stateless (sin estado) y statefull (con estado) por ejemplo map y filter (stateless) y agregaciones como count, group (stateful), joins y windowing (ventanas).
Utiliza estas transformaciones aplicando las reglas de negocios definidas por el Drools. (reglas de negocio)

Los resultados son las salidas del procesamiento realizado por el Kafka aplicando las reglas del Drools.

Para la consulta de los datos procesados utiliza el script del Consumer ejecutando desde el terminal para identificar si todo ha ocurrido bien.

Filtro estudiantil

Al ejecutar el procesamiento de los filtros con las reglas del Drools, podemos visualizar el resultado con el Consumer. Percibimos que los clientes inseridos corresponden con las reglas creadas en el Drools. Los clientes inseridos tienen la edad entre 18-25 años, que están en la universidad y que no tienen ninguna deuda. (Código 22, Código 23)

```
Kins-MacBook-Pro:bin kintat$ ./kafka-console-consumer.sh  \
> --topic FilterStudants1    \
> --from-beginning          \
> --bootstrap-server localhost:9092 \
>
```

Código 22 - Comando visualización del filtro estudiantil.

```
Kins-MacBook-Pro:bin kintat$ ./kafka-console-consumer.sh  \
> --topic FilterStudants1    \
> --from-beginning          \
> --bootstrap-server localhost:9092 \
>
{"id":4,"age":23,"job":"student","marital":"single","education":"university.degree","debtor":"no",
"housing":"yes","loan":"no"}
{"id":6,"age":21,"job":"student","marital":"single","education":"university.degree","debtor":"no",
"housing":"yes","loan":"no"}
{"id":7,"age":23,"job":"student","marital":"single","education":"university.degree","debtor":"no",
"housing":"yes","loan":"no"}
{"id":8,"age":21,"job":"student","marital":"single","education":"university.degree","debtor":"no",
"housing":"yes","loan":"no"}
```

Código 23 - Resultado filtro estudiantil.

Filtros jubilados

Resultado del filtro de los clientes jubilados. Es posible observar que solo fue inserido el cliente con edad 54 años, solteros y que está endeudado. El motivo de traer apenas un elementes es que de acuerdo con el documento de entrada mapeado solo existe apenas un cliente de acuerdo con las reglas de negocio definidas. (Código 24, Código 25)

```
Kins-MacBook-Pro:bin kintat$ ./kafka-console-consumer.sh  \
```

```
> --from-beginning       \
> --topic FilterRetireds1     \
> --bootstrap-server localhost:9092   \
>
```

Código 24 - Comando visualización del filtro Jubilados.

```
Kins-MacBook-Pro:bin kintat$ ./kafka-console-consumer.sh  \
> --from-beginning       \
> --topic FilterRetireds1     \
> --bootstrap-server localhost:9092   \
>
{"id":6,"age":54,"job":"technician","marital":"single","education":"university.degree","debtor":"y
es","housing":"no","loan":"no"}
```

Código 25 - Resultado filtro jubilados.

Filtros adultos

Resultado de los filtros de los clientes adultos. Muestra los clientes que están en el rango 25-45 años que no tengan casa y que están sin deuda. Es posible observar que los resultados está de acuerdo lo que fue definido. (Código 26, Código 27)

```
Kins-MacBook-Pro:bin kintat$ ./kafka-console-consumer.sh  \
> --from-beginning       \
> --topic FilterAdults1     \
> --bootstrap-server localhost:9092   \
>
```

Código 26 - Comando visualización del filtro adultos.

```
ins-MacBook-Pro:bin kintat$ ./kafka-console-consumer.sh  \
> --from-beginning       \
> --topic FilterAdults1     \
> --bootstrap-server localhost:9092   \
```

```
>
{"id":8,"age":37,"job":"services","marital":"married","education":"high.school","debtor":"no","ho
using":"yes","loan":"no"}
{"id":9,"age":29,"job":"blue-
collar","marital":"single","education":"high.school","debtor":"no","housing":"no","loan":"yes"}
{"id":5,"age":35,"job":"blue-
collar","marital":"married","education":"basic.6y","debtor":"no","housing":"yes","loan":"no"}
{"id":5,"age":28,"job":"blue-
collar","marital":"married","education":"basic.6y","debtor":"no","housing":"yes","loan":"no"}
{"id":7,"age":30,"job":"unemployed","marital":"married","education":"high.school","debtor":"no"
,"housing":"no","loan":"no"}
{"id":8,"age":41,"job":"technician","marital":"single","education":"high.school","debtor":"no","ho
using":"yes","loan":"no"}
{"id":10,"age":37,"job":"admin.","marital":"married","education":"high.school","debtor":"no","ho
using":"yes","loan":"no"}
{"id":6,"age":35,"job":"technician","marital":"married","education":"university.degree","debtor":
"no","housing":"no","loan":"yes"}
{"id":7,"age":30,"job":"blue-
collar","marital":"married","education":"university.degree","debtor":"no","housing":"no","loan":"
no"}
{"id":5,"age":34,"job":"services","marital":"married","education":"high.school","debtor":"no","ho
using":"no","loan":"no"}
{"id":10,"age":41,"job":"admin.","marital":"married","education":"university.degree","debtor":"n
o","housing":"yes","loan":"no"}
```


Código 27 - Resultado filtro adultos

Contador de trabajos por clientes

Resultado del contador de trabajos por clientes. Nos trae de en tiempo real la cantidad de clientes por trabajo. Imaginamos un escenario no cual un estudiante cambia de trabajo, de estudiante pasa a ser un administrador, el programa va

automáticamente sustraer un elemento de estudiante y sumar un de administración. (Código 28, Código 29).

```
Kins-MacBook-Pro:bin kintat$ ./kafka-console-consumer.sh      \
> --topic ResultsCountJobs1           \
> --from-beginning                    \
> --bootstrap-server localhost:9092    \
> --property print.key=true           \
> --property value.deserializer=org.apache.kafka.common.serialization.LongDeserializer \
>
```

Código 28 - Comando visualización del contador de trabajos.

```
Kins-MacBook-Pro:bin kintat$ ./kafka-console-consumer.sh      \
> --topic ResultsCountJobs1           \
> --from-beginning                    \
> --bootstrap-server localhost:9092    \
> --property print.key=true           \
> --property value.deserializer=org.apache.kafka.common.serialization.LongDeserializer \
>
management      1
retired         1
blue-collar     2
student         2
services        1
admin.          1
technician      2
management      1
retired         1
blue-collar     2
student         2
services        1
admin.          1
```

```
technician  2
```

Código 29 - Resultado visualización del contador de trabajos.

Filtros de los clientes a contactar

Resultado para volver a contactar los usuarios. Nos trae los resultados de los clientes que hace más de tres meses del último contacto. Por ejemplo, el cliente de id 12, 32, 34, 41, 55 serán contactados caso tenga que hacer alguna acción, por ejemplo. (Código 30, Código 31)

```
Kins-MacBook-Pro:bin kintat$ ./kafka-console-consumer.sh       \
> --topic ClientContactBackContact1       \
> --from-beginning                        \
> --bootstrap-server localhost:9092 \
>
```

Código 30 - Comando visualización de los contactos para volver a contactar.

```
Kins-MacBook-Pro:bin kintat$ ./kafka-console-consumer.sh       \
> --topic ClientContactBackContact1       \
> --from-beginning                        \
> --bootstrap-server localhost:9092 \
>
{"id":12,"contact":"telephone","day":26,"month":"jan","dayOfWeek":"mon","duration":222,"cam
paign":1,"pdays":999,"previous":0,"outcome":"nonexistent"}
{"id":32,"contact":"telephone","day":26,"month":"fev","dayOfWeek":"mon","duration":386,"cam
paign":1,"pdays":999,"previous":0,"outcome":"nonexistent"}
{"id":34,"contact":"telephone","day":29,"month":"fev","dayOfWeek":"mon","duration":230,"cam
paign":1,"pdays":999,"previous":0,"outcome":"nonexistent"}
{"id":40,"contact":"telephone","day":14,"month":"mar","dayOfWeek":"mon","duration":137,"ca
mpaign":1,"pdays":999,"previous":0,"outcome":"nonexistent"}
{"id":41,"contact":"telephone","day":2,"month":"mar","dayOfWeek":"mon","duration":366,"cam
paign":1,"pdays":999,"previous":0,"outcome":"nonexistent"}
```

```
{"id":55,"contact":"telephone","day":28,"month":"fev","dayOfWeek":"mon","duration":269,"campaign":2,"pdays":999,"previous":0,"outcome":"nonexistent"}
```

Código 31 - Resultado de los contactos para volver a contactar.

Contador de duración

Resultado del contador de duración. Resulta la suma de duración de los clientes que en todos los contactos realizados. Es posible utilizar esta información para identificar clientes que están satisfechos con el producto o puede ser los clientes que están demasiadamente insatisfecho por el producto. (Código 32, Código 33)

```
Kins-MacBook-Pro:bin kintat$ ./kafka-console-consumer.sh       \
> --topic ClientContactDuration1          \
> --from-beginning                        \
> --bootstrap-server localhost:9092       \
> --property print.key=true               \
> --property key.deserializer=org.apache.kafka.common.serialization.IntegerDeserializer \
> --property value.deserializer=org.apache.kafka.common.serialization.LongDeserializer \
>
```

Código 32 - Comando visualización del contador de duración.

```
80	208
81	193
82	212
83	165
84	1042
85	20
86	246
87	529
88	192
89	1467
```

```
90	188
91	180
92	48
93	213
94	545
95	583
96	221
97	426
98	287
99	197
100	257
101	229
102	55
```

Código 33 - Resultado contador de duración.

Variación saldo del cliente

Resultado del contador de variación en el saldo del cliente. Podemos observar que los resultados son positivos, o sea, nos devuelve la diferencia entre los sueldos. Ejemplo, tengo un saldo 1000 y tengo el próximo estado del saldo es 10.000, significa que tenemos 9.000 de variación del sueldo, puede ser un señal de examinar mejor este cliente. (Código 34, Código 35)

```
Kins-MacBook-Pro:bin kintat$ ./kafka-console-consumer.sh    \
> --topic ClientEconomicBalance1            \
> --from-beginning                          \
> --bootstrap-server localhost:9092         \
> --property print.key=true                 \
> --property key.deserializer=org.apache.kafka.common.serialization.IntegerDeserializer \
> --property value.deserializer=org.apache.kafka.common.serialization.LongDeserializer \
>
```

Código 34 - Comando visualización del contador variación en el saldo del cliente.

22	779
23	23
24	50
25	0
26	372
27	255
28	113
29	246
30	265
31	839
32	378
33	39
34	0
35	10635
36	63
37	7
38	3
39	506
40	0
41	2586
42	49
43	104
44	529
45	96
46	171
47	364
48	0
49	0
50	0
51	1291
52	244

Código 35 - Resultado contador variación en el saldo del cliente.

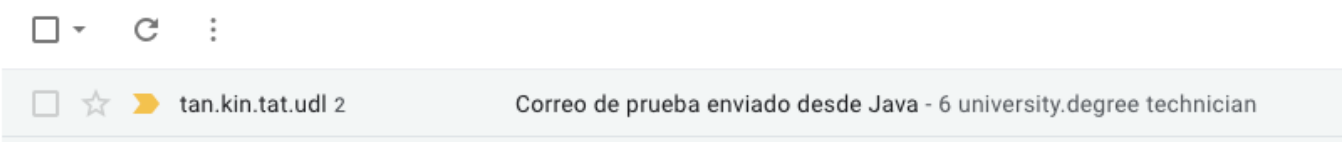
Envío de Email en los clientes potenciales

Con el resultado que hemos logrado en el procesamiento integrando Kafka Stream y Drools, el envío de coreo electrónico es una de las maneras de utilizar el aplicar en herramientas y conceptos de Marketing Directo. Las maneras de ampliar el publico deseado y mantener la fidelidad de los clientes es estar presente y estar siempre al alcance, resolviendo los problemas.

El envío de coreo electrónico es una alternativa para atraer el público para estar al tanto de las novedades. De acuerdo con los resultados obtenidos con el Kafka, fue aplicado una prueba en generar coreos electrónicos en tiempo real, cuando algún cliente es selecciona de acuerdo con las reglas del Drools.
He creado un coreo en el Gmail (coreo electrónico del Google) para el envío de coreos electrónicos a los respectivos clientes.

La Ilustración 18 representa el envío del coreo electrónico con suceso.

Ilustración 18 - Representación del envío de mensajes a través de los resultados del procesado.



La Ilustración 19 representa el cuerdo del coreo. Es el modelo de prueba, es posible cambiar de acuerdo con las necesidades del cliente.

Ilustración 19 - Representación del coreo enviado a los resultados del procesado.

tan.kin.tat.udl@gmail.com
para mí

6

university.degree

technician

Responder Reenviar

4 CONCLUSIONES

Observamos que las herramientas y soluciones de Big Data están tornando cada vez más común hoy en día. La utilización de herramientas de procesamiento streaming (procesamiento en tiempo real) aplicando se en datos de flujo continuo es de grande interés dentro de una empresa, tiene un valor positivo, porque facilita identificar mejor el cliente proporcionando un producto con más calidad y como también identificar los problemas de clientes o oficinas de manera rápida, facilitando su localización para solucionar de manera inmediata, o dependiendo del problema que es necesario de más detalles almacenando los datos mas relevantes para solucionar con auxilio de expertos.

Otra herramienta interesante es el Drools, su integración en definir las reglas de negocio es de mucha importancia y de extremo valor para las empresas, el motivo es el tiempo de hacer una modificación en el código directo por motivos de cambios de reglas de negocio es demasiado costoso. La diferencia entre streaming (registros de cambios) con las tablas estructuradas (database) de hoy en día, es que con los datos estructurados tiene apenas una foto del estado más reciente de los registros, en cambio datos streaming construye los cambios de registros a través de las secuencias de pasos que fueron modificados, eso nos permite utilizar los datos para hacer un filtro temporal de cada, usuario o producto.

En el trabajo realizado puede se concluir que las empresas bancarias tienen fuentes de datos valiosos que necesitan ser trabajados de maneras rápidas para entregar productos adecuados para los clientes promoviendo el marketing financiero utilizando segmentación. Ejemplo, cuando un nuevo cliente se registra, por ejemplo, se puede lanzar de manera rápida, un plano de recomendación de productos que se encajan en su perfil, porque desde un punto de vista del estudiante no tiene sentido tener promociones de planos de jubilados en cambio es necesario recomendar planos de universidad o descuentos a cursos, por ejemplo.

Apache Flink es gran herramienta para el procesamiento, pero la opción por el Kafka Streaming simplemente es debido su fuerte conexión con el Kafka y por la facilidad en integración con las aplicaciones de Java. Los principales problemas en las empresas son en los cambios de reglas de negocio, es decir, los proyectos la parte técnica y las reglas de negocios están muy conectadas, para hacer algún cambio de

una pequeña regla, es necesario activar el equipo técnico para hacer una posible investigación para realizar la modificación, porque generalmente no es solo apenas un lugar que está aplicando esta regla, e si muchos lugares. Si una vez aplicados las reglas de negocio con tablas de decisiones, por ejemplo, facilita mucho el sector de negocios, porque se utiliza el modelo Excel que es más próximo y fácil de modificar sin tener que aprender programación o el mismo lenguaje de reglas del Drools.

Un ejemplo práctico con la salida del procesamiento fue realizado, el envío de coreo electrónico para los clientes que fueron seleccionados, tiene el objetivo de mostrar una aplicación básica del marketing directo que es promover propagandas directo, pero seleccionando un cliente de un público específico correcto, no habiendo correos para todos los grupos con emails generales por defecto.

La integración de Drools con herramientas de Big Data como el Kafka, es muy interesante primero para encontrar una solución para el problema de procesamiento en tiempo real como también para el aprendizaje de las nuevas herramientas que están creciendo en el sector de tecnología.

5 TRABAJO FUTURO

Este trabajo fue utilizado las características básicas del Drools para integración con el Kafka. El Drools tiene diversas funciones que facilita la organización de las tareas y reglas en una aplicación, ayudando el sector de negocios a gestionar con simplicidad las reglas de negocio. Como trabajo futuro, una de las funcionalidades del Drools es el Workbench, esta funcionalidad posibilita utilizar una aplicación web para gestionar con más facilidad las reglas de negocio de una manera en general, como creación, modificación. Su integración es importante porque auxiliar en las tareas de una visión del usuario de negocios sin tocar demasiado la parte de código y con mucha mas facilidad.

Otra mejoría es utilizar algún banco de datos no estructurados para el almacenamiento de los datos de salida que serán procesados del Kafka Streaming en tiempo real. El motivo para esta mejoría es que futuramente es posible utilizar los datos procesados hacer una investigación con más particularidad separando en los conjuntos de acuerdo con los perfiles de cada usuario, o utilizar para conectar con otras fuentes de datos. Cassandra es una base de datos NoSQL capaz de

almacenar los datos de clave y valor escrita en Java que permite en almacenar grande volúmenes de datos en forma distribuida, o el MongoDB que permite guardar en una estructuras de datos BSON (una especificación similar a JSON) con un esquema dinámico. La principal ventaja de los bancos de datos no estructurados es su crecimiento un ritmo exponencial en el almacenamiento de los datos, porque suportan diferentes estructuras de datos sin incrementar la complejidad en su manejo , capacidad de manejar grandes volúmenes de información provenientes desde diferentes fuentes de información, acopla al desarrollo ágil, incrementando en diversos sistemas diferentes y etc.

Es necesario pensar y desarrollar maneras de marketing directo con los datos procesados, este trabajo fue utilizado el envío de coreo como una demonstración de aplicación real de acuerdo con los datos procesados. Hay muchas maneras en utilizar el marketing directo, como por ejemplo encontrar grupos por geolocalización para compañas publicitaria., descuento de productos de acuerdo con código de promoción, etc.

6 REFERENCIAS BIBLIOGRÁFICAS

GRAHAM, IAN**. Business Rules Management and Service Oriented** Architecture A Pattern Language. 1.ed. Local: England, 2006. 274 p.

MARTIN IBARRA, GONZALO; BAZÁN, PATRICIA; **Análisis y comparación de plataformas BRMS a través de una prueba de concepto.** Disponible en **<https://www.linti.unlp.edu.ar/uploads/docs/analisis_y_comparacion_de_plataformas_brms_a_traves_de_una_prueba_de_concepto.pdf>.** Acceso en: 21 abr. 2019.

ARCE MARTÍNEZ, DANIEL; **Transformación de la forma tradicional de banca hacia el mundo digital.** Disponible en: <http://repositorio.upct.es/bitstream/handle/10317/5900/tfg-arc-tra.pdf>. Acceso en: 21 abr. 2019.

GARCÍA MONTALVO, JOSÉ; **El impacto del "Big data" en los servicios financieros.** Disponible en: <http://www.econ.upf.edu/~montalvo/wp/big_data_banking_v4.pdf>. Acceso en: 21 abr. 2019.

OCAÑA, CARLOS; URÍA, FRANCISCO. **El nivel de madurez digital.** Disponible en: <https://www.funcas.es/publicaciones/docs/informe01.pdf>. Acceso en: 14 may. 2019.

Retail banks and big data: Big data as the key to better risk management. Disponible en: <https://eiuperspectives.economist.com/sites/default/files/RetailBanksandBigData.pdf>. Acceso en 14 may.

KOTESHOV, DMITRI. **FRAUD MANAGEMENT: DETECTION AND PREVENTION IN BANKING INDUSTRY**. Disponible en: <https://www.elinext.com/blog/fraud-management-detection-and-prevention-in-banking-industry/>. Acceso en 14 may.

BAESENS, BART; VAN VLASSELAER, VÉRONIQUE; VERBEKE, WOUTER. **FRAUD ANALYTICS** Using descriptive, predictive and social network techniques. Disponible en: <http://www.dataminingapps.com/wp-content/uploads/2015/08/68614_excerpt-1.pdf>. Acceso en 14 may. 2019.

DUMOULIN, MATHIEU. **Better Complex Event Processing at Scale Using a Microservices-based Streaming Architecture**. Disponible en: <https://mapr.com/blog/better-complex-event-processing-scale-using-microservices-based-streaming-architecture-part-1/> Acceso en 14 may. 2019.

MURALI KRISHNA, KUMAR BARUAH, PALLAV. **High Performance Kafka Powered Scalable Real Time Rule Engine Model for Event Stream Processing.** Disponible en: <https://www.ijser.org/researchpaper/High-Performance-Kafka-Powered-Scalable-Real-Time-Rule-Engine-Model-for-Event-Stream-Processing.pdf> Acceso en 14 may. 2019.

VASEEJARAN, GOWTHAMY. **A Gentle Introduction to Stream Processing** Disponible en: <https://medium.com/@gowthamy/big-data-battle-batch-processing-vs-stream-processing-5d94600d8103>. Acceso en 21 abr. 2019.

MARTIN ORTIZ, NATALIA. **Big data en el sector bancario: análisis y planteamiento de una línea de servicios.** Disponible en: <http://oa.upm.es/42936/1/TFM_NATALIA_MARTIN_ORTIZ.pdf>. Acceso en 27 abr. 2019.

CÓRDOBA, GUILLERMO. **Segmentación de clientes: Algunos ejemplos prácticos.** Disponible en: <https://www.unica360.com/segmentacion-de-clientes-una-propuesta-de-clasificacion-i>. Acceso en 28 abr. 2019.

VIDAL, JORDI. **El ciclo de vida del cliente. Un paso previo a la segmentación estratégica en el sector bancario.** Disponible en: <https://www.unica360.com/el-ciclo-de-vida-del-cliente-un-paso-previo-a-la-segmentacion-estadistica-en-el-sector-bancario>. Acceso en 28 abr. 2019.

MENDELSON, HAIM. **Reinventar la empresa** en la era digital. Disponible en: <https://www.bbvaopenmind.com/wp-content/uploads/2015/02/BBVA-OpenMind-modelos-de-negocio-tecnologias-de-la-informacion-y-la-empresa-del-futuro-Haim-Mendelson.pdf.pdf>. Acceso en 14 may. 2019.

RIVERA CAMINO, JAIME; DE GARCILLÁN LÓPEZ-RUA, MENCÍA**. Marketing sectorial. Principios y aplicaciones**. 1.ed. Local: Madrid, 2014. 390 p.

BOUBETA PUIG, JUAN; ORTIZ BELLOT, GUADALUPE; MEDINA BULO, INMACULADA. **Procesamiento de Eventos Complejos en Entornos SOA: Caso**

de Estudio para la Detección Temprana de Epidemias <http://lbd.udc.es/jornadas2011/actas/JCIS/JCIS/S3/S3_1_paper.pdf>. Acceso en 30 abr. 2019.

AYLLÓN, VICTOR; M. REINA, JUAN. **CEP/ESP: Procesamiento y correlación de gran cantidad de eventos en arquitectura SOA**. <https://www.isa.us.es/downloads/proceedings/0097.pdf>. Acceso en 30 abr. 2019.

MENDELSON, HAIM. **Modelos de negocios, tecnologías de la información y la empresa del futuro**. <https://www.bbvaopenmind.com/wp-content/uploads/2015/02/BBVA-OpenMind-modelos-de-negocio-tecnologias-de-la-informacion-y-la-empresa-del-futuro-Haim-Mendelson.pdf.pdf>. Acceso en 30 abr. 2019.

JARAMILLO, JAVIER PEÑAS; SAINZ, LAURA REYERO. **JLOP**. <https://eprints.ucm.es/16693/1/Memoria_JLOPfinal.pdf>. Acceso en 13 may. 2019.

SANCHES, CARLOS CARRILLO. **BIG DATA: ANÁLISIS Y ESTUDIO DE LA PLATAFORMA HADOOP**. <http://oa.upm.es/34779/1/PFC_noelia_jimenez_barquin.pdf>. Acceso en 13 may. 2019.

PROGRESS. **5 REASONS TO ADOPT BUSINESS RULES.** Disponible en: <https://www.progress.com/docs/default-source/default-document-library/Progress/Documents/Papers/PROGRESS_TT_WP_FINAL.pdf>. Acceso en 13 may. 2019.

MEYER, CHRIS; SANDS, LITA; SEYFERT-MARGOLIS, VICKI; BUNGART, STEFAN; C. HERZ, J. **Big data El poder de los datos.** Disponible en: <https://www.fundacionbankinter.org/documents/20183/42758/Publicaci%C3%B3n+Big+data/cc4bd4e9-8c9b-4052-8814-ccbd48324147>. Acceso en 14 may. 2019.

PORRAS CASTAÑO, JAVIER. **Big Data en el Sector Financiero.** Disponible en: <https://www.revistabyte.es/actualidad-byte/big-data-sector-financiero-2/>. Acceso en 14 may. 2019.

ANDRÉS BLANCO, TERESA. **Las operaciones bancarias digitales crecen en BBVA.** Disponible en: <https://www.bbva.com/es/operaciones-bancarias-digitales-crecen-bbva/>. Acceso en 17 may. 2019.

LÓPEZ GARCÍA, DAVID. **Análisis de las posibilidades de uso de Big Data en las organizaciones.** Disponible en: <https://repositorio.unican.es/xmlui/bitstream/handle/10902/4528/TFM%20-%20David%20L%C3%B3pez%20Garc%C3%ADaS.pdf?sequence=1>. Acceso en 17 may. 2019.

FREEMAN, HARRINE. **Streaming Analytics 101: The What, Why, and How.** Disponible en: <https://www.dataversity.net/streaming-analytics-101/#>. Acceso en 17 may. 2019.

GONZÁLEZ, TERESA. **¿Qué es el marketing directo? Ventajas, canales y ejemplos.** Disponible en: <https://es.sendinblue.com/blog/marketing-directo/>. Acceso en 17 jun, 2019.

7 ANEXO

RuleSet	rules
Import	Model.Client, Producers.KafkaRuleClientProducer, org.apache.kafka.clients.producer.KafkaProducer, org.apache.kafka.clients.producer.ProducerConfig, org.apache.kafka.clients.producer.ProducerRecord, org.apache.kafka.common.KafkaException, org.apache.kafka.common.serialization.IntegerSerializer, org.apache.kafka.common.serialization.StringDeserializer, org.apache.kafka.common.serialization.StringSerializer, org.apache.logging.log4j.LogManager, org.apache.logging.log4j.Logger, java.util.Properties, Model.Constants

RuleTable Discount rates

NAME	**CONDITION**	**CONDITION**	**CONDITION**	**CONDITION**	**CONDITION**	**CONDITION**	**ACTION**
	$b: Client						
	getAge() > $1, getAge() < $2	getMarital() == $param	getJob() == $param	getHousing() == $param	getEducation() == $param	getDebtor() == $param	KafkaRuleClientProducer.addRuleDroolsClient("KafkaPerfilEstudantil", $param, $b);
NAME	Customer type						Discount
Individual > 3y	18, 25		"student"		"university.degree"	"no"	"FilterStudants90"
Individual < 3y	50, 100	"single"		"no"		"yes"	"FilterRetireds90"
Business Any	25, 45					"no"	"FilterAdults90"